nbsphinx

Release 0.5.1-27-g1ac5b88

Matthias Geier

2020-03-30

Contents

1 Installation 4
  1.1 nbsphinx Packages ................................................. 4
  1.2 nbsphinx Prerequisites ........................................... 6
    1.2.1 Python ....................................................... 6
    1.2.2 Sphinx ...................................................... 6
    1.2.3 pip ........................................................... 7
    1.2.4 pandoc ....................................................... 7
    1.2.5 Pygments Lexer for Syntax Highlighting ..................... 7
    1.2.6 Jupyter Kernel .............................................. 8

2 Usage 8
  2.1 Sphinx Setup ................................................... 8
    2.1.1 Sphinx Configuration Values ............................... 9
      2.1.1.1 exclude_patterns ..................................... 9
      2.1.1.2 extensions ........................................... 9
      2.1.1.3 highlight_language ................................... 9
      2.1.1.4 html_scaled_image_link ................................ 10
      2.1.1.5 html_sourcelink_suffix ................................ 10
      2.1.1.6 latex_additional_files ................................ 10
      2.1.1.7 mathjax_config ....................................... 10
    2.1.2 nbsphinx Configuration Values ............................. 10
      2.1.2.1 nbsphinx_allow_errors ................................ 10
      2.1.2.2 nbsphinx_codecell_lexer ................................ 10
      2.1.2.3 nbsphinx_custom_formats ................................ 11
      2.1.2.4 nbsphinx_epilog ....................................... 11
      2.1.2.5 nbsphinx_execute ..................................... 11
      2.1.2.6 nbsphinx_execute_arguments ........................... 11
      2.1.2.7 nbsphinx_input_prompt ................................ 11
      2.1.2.8 nbsphinx_kernel_name .................................. 11
      2.1.2.9 nbsphinx_output_prompt ................................ 12
      2.1.2.10 nbsphinx_prolog ...................................... 12
      2.1.2.11 nbsphinx_prompt_width ................................ 12
      2.1.2.12 nbsphinx_requirejs_options .......................... 12
      2.1.2.13 nbsphinx_requirejs_path .............................. 12
      2.1.2.14 nbsphinxponsive_width ............................... 12
      2.1.2.15 nbsphinx_thumbnails .................................. 12
      2.1.2.16 nbsphinx_timeout ..................................... 13
      2.1.2.17 nbsphinx_widgets_options ............................ 13
nbsphinx is a Sphinx\(^1\) extension that provides a source parser for *.ipynb files. Custom Sphinx directives are used to show Jupyter Notebook\(^2\) code cells (and of course their results) in both HTML and LaTeX output. Un-evaluated notebooks – i.e. notebooks without stored output cells – will be automatically executed during the Sphinx build process.

**Quick Start:**

1. Install nbsphinx
2. Edit your `conf.py` and add `nbsphinx` to `extensions`.
3. Edit your `index.rst` and add the names of your *.ipynb files to the `toctree`.
4. Run Sphinx!

**Online documentation (and example of use):** [http://nbsphinx.readthedocs.io/](http://nbsphinx.readthedocs.io/)

**Source code repository (and issue tracker):** [https://github.com/spatialaudio/nbsphinx/](https://github.com/spatialaudio/nbsphinx/)

**License:** MIT – see the file LICENSE for details.

All content shown below – except for the sections Normal reStructuredText Files (page 54), Contributing (page 61) and Version History (page 62) – was generated from Jupyter notebooks.

---

## 1 Installation

Note that some packages may be out of date. You can always get the newest nbsphinx release from PyPI\(^3\) (using pip). If you want to try the latest development version, have a look at the section Contributing (page 61).

### 1.1 nbsphinx Packages

**Anaconda Cloud 0.5.1**

If you are using the conda package manager (e.g. with Anaconda\(^5\) for Linux/macOS/Windows), you can install nbsphinx from the conda-forge\(^6\) channel:

```
conda install -c conda-forge nbsphinx
```

If you are using Linux, there are packages available for many distributions.

---

\(^1\) [https://www.sphinx-doc.org/](https://www.sphinx-doc.org/)
\(^2\) [https://jupyter.org/](https://jupyter.org/)
\(^3\) [https://pypi.org/project/nbsphinx](https://pypi.org/project/nbsphinx)
\(^4\) [https://anaconda.org/conda-forge/nbsphinx](https://anaconda.org/conda-forge/nbsphinx)
\(^5\) [https://www.anaconda.com/distribution/](https://www.anaconda.com/distribution/)
\(^6\) [https://conda-forge.org/](https://conda-forge.org/)
<table>
<thead>
<tr>
<th>Package</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALT Linux p9</td>
<td>0.2.15</td>
</tr>
<tr>
<td>ALT Sisyphus</td>
<td>0.2.15</td>
</tr>
<tr>
<td>AUR</td>
<td>0.5.0</td>
</tr>
<tr>
<td>Debian Oldstable</td>
<td>0.2.9</td>
</tr>
<tr>
<td>Debian Stable</td>
<td>0.4.2</td>
</tr>
<tr>
<td>Debian Testing</td>
<td>0.4.3</td>
</tr>
<tr>
<td>Debian Unstable</td>
<td>0.4.3</td>
</tr>
<tr>
<td>Deepin</td>
<td>0.3.1</td>
</tr>
<tr>
<td>Devuan 2.0 (ASCII)</td>
<td>0.2.9</td>
</tr>
<tr>
<td>Devuan 3.0 (Beowulf)</td>
<td>0.4.2</td>
</tr>
<tr>
<td>Devuan 4.0 (Chimaera)</td>
<td>0.4.3</td>
</tr>
<tr>
<td>Devuan Unstable</td>
<td>0.4.3</td>
</tr>
<tr>
<td>Fedora 26</td>
<td>0.2.13</td>
</tr>
<tr>
<td>Fedora 27</td>
<td>0.2.13</td>
</tr>
<tr>
<td>Fedora 28</td>
<td>0.2.17</td>
</tr>
<tr>
<td>Fedora 29</td>
<td>0.2.17</td>
</tr>
<tr>
<td>Fedora 30</td>
<td>0.4.2</td>
</tr>
<tr>
<td>Fedora 31</td>
<td>0.4.2</td>
</tr>
<tr>
<td>Fedora 32</td>
<td>0.4.2</td>
</tr>
<tr>
<td>Fedora Rawhide</td>
<td>0.4.2</td>
</tr>
<tr>
<td>Funtoo 1.3</td>
<td>0.2.14</td>
</tr>
<tr>
<td>Funtoo 1.4</td>
<td>0.4.1</td>
</tr>
<tr>
<td>Gentoo</td>
<td>0.4.1</td>
</tr>
<tr>
<td>Kali Linux Rolling</td>
<td>0.4.3</td>
</tr>
<tr>
<td>nixpkgs stable</td>
<td>0.4.2</td>
</tr>
<tr>
<td>nixpkgs unstable</td>
<td>0.5.1</td>
</tr>
<tr>
<td>openSUSE Leap 15.0</td>
<td>0.3.3</td>
</tr>
<tr>
<td>openSUSE Leap 15.1</td>
<td>0.3.3</td>
</tr>
<tr>
<td>openSUSE Leap 15.2</td>
<td>0.5.1</td>
</tr>
<tr>
<td>openSUSE Tumbleweed</td>
<td>0.5.1</td>
</tr>
<tr>
<td>Pardus</td>
<td>0.2.9</td>
</tr>
<tr>
<td>Parrot</td>
<td>0.4.3</td>
</tr>
<tr>
<td>PureOS Amber</td>
<td>0.4.2</td>
</tr>
<tr>
<td>PureOS landing</td>
<td>0.4.3</td>
</tr>
<tr>
<td>Raspbian Oldstable</td>
<td>0.2.9</td>
</tr>
<tr>
<td>Raspbian Stable</td>
<td>0.4.2</td>
</tr>
<tr>
<td>Raspbian Testing</td>
<td>0.4.3</td>
</tr>
<tr>
<td>Ubuntu 18.04</td>
<td>0.3.1</td>
</tr>
<tr>
<td>Ubuntu 19.04</td>
<td>0.4.2</td>
</tr>
<tr>
<td>Ubuntu 19.10</td>
<td>0.4.2</td>
</tr>
<tr>
<td>Ubuntu 20.04</td>
<td>0.4.3</td>
</tr>
</tbody>
</table>

On any platform, you can also install nbsphinx with `pip`, Python’s own package manager:

```
python3 -m pip install nbsphinx --user
```

If you want to install it system-wide for all users (assuming you have the necessary rights), just drop the `--user` flag.

To upgrade an existing nbsphinx installation to the newest release, use the `--upgrade` flag:

```
python3 -m pip install nbsphinx --upgrade --user
```

If you suddenly change your mind, you can un-install it with:

```
python3 -m pip uninstall nbsphinx
```

Depending on your Python installation, you may have to use `python` instead of `python3`.

### 1.2 nbsphinx Prerequisites

Some of the aforementioned packages will install some of these prerequisites automatically, some of the things may be already installed on your computer anyway.

#### 1.2.1 Python

Of course you’ll need Python, because both Sphinx and nbsphinx are implemented in Python. There are many ways to get Python. If you don’t know which one is best for you, you can try Anaconda⁹.

#### 1.2.2 Sphinx

You’ll need Sphinx¹⁰ as well, because nbsphinx is just a Sphinx extension and doesn’t do anything on its own.

If you use conda, you can get Sphinx from the conda-forge channel¹¹:

```
conda install -c conda-forge sphinx
```

Alternatively, you can install it with pip (see below):

```
python3 -m pip install Sphinx --user
```

---

⁸ https://pypi.org/project/nbsphinx  
⁹ https://www.anaconda.com/distribution/  
¹⁰ https://www.sphinx-doc.org/  
¹¹ https://anaconda.org/conda-forge/sphinx
1.2.3 pip

Recent versions of Python already come with pip pre-installed. If you don’t have it, you can install it manually\(^\text{12}\).

1.2.4 pandoc

The stand-alone program pandoc\(^\text{13}\) is used to convert Markdown content to something Sphinx can understand. You have to install this program separately, ideally with your package manager. If you are using conda, you can install pandoc from the conda-forge channel\(^\text{14}\):

```
conda install -c conda-forge pandoc
```

If that doesn’t work out for you, have a look at pandoc’s installation instructions\(^\text{15}\).

Note

The use of pandoc in nbsphinx is temporary, but will likely stay that way for a long time, see issue \#36\(^\text{16}\).

1.2.5 Pygments Lexer for Syntax Highlighting

To get proper syntax highlighting in code cells, you’ll need an appropriate Pygments lexer. This of course depends on the programming language of your Jupyter notebooks (more specifically, the pygments lexer metadata of your notebooks).

For example, if you use Python in your notebooks, you’ll have to have the IPython package installed, e.g. with

```
conda install -c conda-forge ipython
```

or

```
python3 -m pip install IPython --user
```

Note

If you are using Anaconda with the default channel and syntax highlighting in code cells doesn’t seem to work, you can try to install IPython from the conda-forge channel or directly with pip, or as a work-around, add ‘IPython.sphinxext.ipython_console_highlighting’ to extensions in your conf.py.

For details, see Anaconda issue \#1430\(^\text{17}\) and nbsphinx issue \#24\(^\text{18}\).

\(^{13}\) https://pandoc.org/
\(^{14}\) https://anaconda.org/conda-forge/pandoc
\(^{15}\) https://pandoc.org/installing.html
\(^{16}\) https://github.com/spatialaudio/nbsphinx/issues/36
\(^{17}\) https://github.com/ContinuumIO/anaconda-issues/issues/1430
\(^{18}\) https://github.com/spatialaudio/nbsphinx/issues/24
1.2.6 Jupyter Kernel

If you want to execute your notebooks during the Sphinx build process (see Controlling Notebook Execution (page 39)), you need an appropriate Jupyter kernel\(^\text{19}\) installed.

For example, if you use Python, you should install the ipykernel package, e.g. with

```
conda install -c conda-forge ipykernel
```

or

```
python3 -m pip install ipykernel --user
```

If you created your notebooks yourself with Jupyter, it’s very likely that you have the right kernel installed already.

The following section was generated from `doc/usage.ipynb`.

2 Usage

2.1 Sphinx Setup

In the directory with your notebook files, run this command (assuming you have Sphinx\(^\text{20}\) installed already):

```
python3 -m sphinx.cmd.quickstart
```

Answer the questions that appear on the screen. In case of doubt, just press the `<Return>` key repeatedly to take the default values.

After that, there will be a few brand-new files in the current directory. You’ll have to make a few changes to the file named `conf.py`. You should at least check if those two variables contain the right things:

```python
extensions = [
    'nbsphinx',
    'sphinx.ext.mathjax',
]
exclude_patterns = ['_build', '**.ipynb_checkpoints']
```

For an example, see this project’s `conf.py` file.

Once your `conf.py` is in place, edit the file named `index.rst` and add the file names of your notebooks (without the `.ipynb` extension) to the `toctree`\(^\text{21}\) directive. For an example, see this project’s `doc/index.rst` file.


\(^\text{20}\) https://www.sphinx-doc.org/

2.1.1 Sphinx Configuration Values

All configuration values are described in the Sphinx documentation\(^22\), here we mention only the ones which may be relevant in combination with nbsphinx.

2.1.1.1 exclude_patterns

Sphinx builds all potential source files (reST files, Jupyter notebooks, …) that are in the source directory, whether they are linked to or not. If you want certain files not to be built, specify them in `exclude_patterns`\(^23\). For example, you might want to ignore files in your build directory and temporary Jupyter files:

```python
exclude_patterns = [ '_build', '**.ipynb_checkpoints' ]
```

2.1.1.2 extensions

This is the only required value. You have to add `nbsphinx` to the list of `extensions`\(^24\), otherwise it won’t work.

Other interesting extensions are:

- `sphinx.ext.mathjax` for math formulas (page 19)
- `sphinxcontrib.bibtex` for bibliographic references (page 57)
- `sphinxcontrib.rsvgconverter` for SVG->PDF conversion in LaTeX output (page 22)
- `sphinx_copybutton` for adding “copy to clipboard” buttons\(^25\) to all text/code boxes
- `sphinx_gallery.load_style` to load CSS styles for thumbnail galleries (page 48)

2.1.1.3 highlight_language

Default language for syntax highlighting in reST and Markdown cells, when no language is specified explicitly.

By default, this is `python3`, while Jupyter doesn’t have a default language. Set `highlight_language`\(^26\) to `none` to get the same behavior as in Jupyter:

```python
highlight_language = 'none'
```

See also `nbsphinx_codecell_lexer` (page 10).


\(^{25}\) [https://sphinx-copybutton.readthedocs.io/](https://sphinx-copybutton.readthedocs.io/)

2.1.1.4 html_scaled_image_link

As a work-around – until https://github.com/sphinx-doc/sphinx/issues/4229 is solved – you should set html_scaled_image_link\(^{27}\) to False:

```
html_scaled_image_link = False
```

2.1.1.5 html_sourcelink_suffix

By default, a .txt suffix is added to source files. This is only relevant if the chosen HTML theme supports source links and if html_show_sourcelink\(^{28}\) is True.

Jupyter notebooks with the suffix .ipynb.txt are normally not very useful, so if you want to avoid the additional suffix, set html_sourcelink_suffix\(^{29}\) to the empty string:

```
html_sourcelink_suffix = ''
```

2.1.1.6 latex_additional_files

latex_additional_files\(^{30}\) can be useful if you are using BibTeX files, see References (page 57).

2.1.1.7 mathjax_config

The configuration value mathjax_config\(^{31}\) can be useful to enable Automatic Equation Numbering (page 20).

2.1.2 nbsphinx Configuration Values

2.1.2.1 nbsphinx_allow_errors

If True, the build process is continued even if an exception occurs.

See Ignoring Errors (page 41).

2.1.2.2 nbsphinx_codecell_lexer

Default Pygments lexer for syntax highlighting in code cells. If available, this information is taken from the notebook metadata instead.

Please note that this is not the same as highlight_language (page 9), which is used for formatting code in Markdown cells!

---


2.1.2.3 nbsphinx_custom_formats

See Custom Notebook Formats (page 46).

2.1.2.4 nbsphinx_epilog

See Prolog and Epilog (page 44).

2.1.2.5 nbsphinx_execute

Whether to execute notebooks before conversion or not. Possible values: 'always', 'never', 'auto' (default).

See Explicitly Dis-/Enabling Notebook Execution (page 40).

2.1.2.6 nbsphinx.ExecuteArguments

Kernel arguments used when executing notebooks.

If you use Matplotlib for plots (page 30), this setting is recommended:

```python
nbsphinx_execute_arguments = [
    "--InlineBackend.figure_formats={'svg', 'pdf'}",
    "--InlineBackend.rc={'figure.dpi': 96},
]
```

If you don't use LaTeX/PDF output, you can drop the 'pdf' figure format.

See Configuring the Kernels (page 43).

2.1.2.7 nbsphinx_input_prompt

Input prompt for code cells. `%s` is replaced by the execution count.

To get a prompt similar to the Classic Notebook, use

```python
nbsphinx_input_prompt = 'In [%s]:'
```

2.1.2.8 nbsphinx_kernel_name

Use a different kernel than stored in the notebook metadata, e.g.:

```python
nbsphinx_kernel_name = 'python3'
```

See Configuring the Kernels (page 42).
2.1.2.9 nbsphinx_output_prompt

Output prompt for code cells. \%s is replaced by the execution count.
To get a prompt similar to the Classic Notebook, use

```
nbsphinx_output_prompt = 'Out[%s]:'
```

2.1.2.10 nbsphinx_prolog

See Prolog and Epilog (page 44).

2.1.2.11 nbsphinx_prompt_width

Width of input/output prompts (HTML only).
Any CSS length can be specified.

2.1.2.12 nbsphinx_requirejs_options

Options for loading RequireJS. See nbsphinx_requirejs_path (page 12).

2.1.2.13 nbsphinx_requirejs_path

URL or local path to override the default URL for RequireJS\(^{32}\).
If you use a local file, it should be located in a directory listed in html_static_path\(^{33}\).
Set to empty string to disable loading RequireJS.

2.1.2.14 nbsphinx_responsive_width

If the browser window is narrower than this, input/output prompts are on separate lines (HTML only).
Any CSS length can be specified.

2.1.2.15 nbsphinx_thumbnails

A dictionary mapping from a document name (i.e. source file without suffix but with subdirectories) – optionally containing wildcards – to a thumbnail path to be used in a thumbnail gallery (page 48).
See Specifying Thumbnails (page 52).

\(^{32}\) https://requirejs.org/
\(^{33}\) http://www.sphinx-doc.org/en/master/usage/configuration.html#confval-html_static_path
2.1.2.16 nbsphinx_timeout

Controls when a cell will time out. The timeout is given in seconds. Given -1, cells will never time out, which is also the default.
See Cell Execution Timeout (page 44).

2.1.2.17 nbsphinx_widgets_options

Options for loading Jupyter widgets resources. See nbsphinx_widgets_path (page 13).

2.1.2.18 nbsphinx_widgets_path

URL or local path to override the default URL for Jupyter widgets resources. See Interactive Widgets (HTML only) (page 34).
If you use a local file, it should be located in a directory listed in html_static_path34.
For loading the widgets resources, RequireJS is needed, see nbsphinx_requirejs_path (page 12).
If nbsphinx_widgets_path is not specified, widgets resources are only loaded if at least one notebook actually uses widgets. If you are loading the relevant JavaScript code by some other means already, you can set this option to the empty string to avoid loading it a second time.

2.2 Running Sphinx

To create the HTML pages, use this command:

```
python3 -m sphinx <source-dir> <build-dir>
```

If you have many notebooks, you can do a parallel build by using the -j option:

```
python3 -m sphinx <source-dir> <build-dir> -j<number-of-processes>
```

For example, if your source files are in the current directory and you have 4 CPU cores, you can run this:

```
python3 -m sphinx . _build -j4
```

Afterwards, you can find the main HTML file in _build/index.html.
Subsequent builds will be faster, because only those source files which have changed will be re-built.
To force re-building all source files, use the -E option.

Note

By default, notebooks will be executed during the Sphinx build process only if they do not have any output cells stored. See Controlling Notebook Execution (page 39).

To create LaTeX output, use:

```
python3 -m sphinx <source-dir> <build-dir> -b latex
```

34 http://www.sphinx-doc.org/en/master/usage/configuration.html#confval-html_static_path
If you don’t know how to create a PDF file from the LaTeX output, you should have a look at Latexmk\textsuperscript{35} (see also this tutorial\textsuperscript{36}).

Sphinx can automatically check if the links you are using are still valid. Just invoke it like this:

```
python3 -m sphinx <source-dir> <build-dir> -b linkcheck
```

### 2.3 Watching for Changes with sphinx-autobuild

If you think it’s tedious to run the Sphinx build command again and again while you make changes to your notebooks, you’ll be happy to hear that there is a way to avoid that: sphinx-autobuild\textsuperscript{37}!

It can be installed with

```
python3 -m pip install sphinx-autobuild --user
```

You can start auto-building your files with

```
python3 -m sphinx_autobuild <source-dir> <build-dir>
```

This will start a local webservice which will serve the generated HTML pages at http://localhost:8000/. Whenever you save changes in one of your notebooks, the appropriate HTML page(s) will be re-built and when finished, your browser view will be refreshed automagically. Neat!

You can also abuse this to auto-build the LaTeX output:

```
python3 -m sphinx_autobuild <source-dir> <build-dir> -b latex
```

However, to auto-build the final PDF file as well, you’ll need an additional tool. Again, you can use latexmk for this (see above (page 13)). Change to the build directory and run

```
latexmk -pdf -pvc
```

If your PDF viewer isn’t opened because of LaTeX build errors, you can use the command line flag \texttt{-f} to force creating a PDF file.

### 2.4 Automatic Creation of HTML and PDF output on readthedocs.org

There are two different methods, both of which are described below.

In both cases, you’ll first have to create an account on https://readthedocs.org/ and connect your GitLab/Github/Bitbucket/… account. Instead of connecting, you can also manually add any publicly available Git/Subversion/Mercurial/Bazaar/… repository.

After doing the steps described below, you only have to “push” to your repository, and the HTML pages and the PDF file of your stuff are automagically created on readthedocs.org. Awesome!

You can even have different versions of your stuff, just use Git tags and branches and select in the readthedocs.org settings\textsuperscript{38} which of those should be created.

---

**Note**

\textsuperscript{35} http://personal.psu.edu/jcc8/software/latexmk-jcc/

\textsuperscript{36} https://mg.readthedocs.io/latexmk.html

\textsuperscript{37} https://pypi.org/project/sphinx-autobuild

\textsuperscript{38} https://readthedocs.org/dashboard/
If you want to execute notebooks (see *Controlling Notebook Execution* (page 39)), you’ll need to install the appropriate Jupyter kernel. In the examples below, the IPython kernel is installed from the packet ipykernel.

### 2.4.1 Using requirements.txt

1. Create a file named `.readthedocs.yml` in the main directory of your repository with the following contents:

```yaml
version: 2
formats: all
python:
  version: 3
  install:
    - requirements: doc/requirements.txt
system_packages: true
```

For further options see https://docs.readthedocs.io/en/latest/config-file/.

2. Create a file named `doc/requirements.txt` (or whatever you chose in the previous step) containing the required `pip` packages:

```plaintext
ipykernel
nbsphinx
```

You can also install directly from Github et al., using a specific branch/tag/commit, e.g.

```plaintext
git+https://github.com/spatialaudio/nbsphinx.git@master
```

### 2.4.2 Using conda

1. Create a file named `.readthedocs.yml` in the main directory of your repository with the following contents:

```yaml
version: 2
formats: all
conda:
  file: doc/environment.yml
```

For further options see https://docs.readthedocs.io/en/latest/config-file/.

2. Create a file named `doc/environment.yml` (or whatever you chose in the previous step) describing a conda environment like this:

```plaintext
channels:
  - conda-forge
dependencies:
  - python>=3
  - pandoc
  - ipykernel
  - pip
  - pip:
    - nbsphinx
```

---

It is up to you if you want to install `nbsphinx` with `conda` or with `pip` (but note that the `conda` package might be outdated). And you can of course add further `conda` and `pip` packages. You can also install packages directly from Github et al., using a specific branch/tag/commit, e.g.

```
- pip:
  - git+https://github.com/spatialaudio/nbsphinx.git@master
```

Note

The specification of the `conda-forge` channel is recommended because it tends to have more recent package versions than the default channel.

2.5 HTML Themes

The `nbsphinx` extension does not provide its own theme, you can use any of the available themes or create a custom one\(^{40}\), if you feel like it.

The following (incomplete) list of themes contains up to three links for each theme:

1. The documentation (or the official sample page) of this theme (if available; see also the documentation of the built-in Sphinx themes\(^{41}\))
2. How the `nbsphinx` documentation looks when using this theme
3. How to enable this theme using either `requirements.txt` or `readthedocs.yml` and theme-specific settings (in some cases)

2.5.1 Sphinx's Built-In Themes

- agogo: example\(^{42}\), usage\(^{43}\)
- alabaster\(^{44}\): example\(^{45}\), usage\(^{46}\)
- bizstyle: example\(^{47}\), usage\(^{48}\)
- classic: example\(^{49}\), usage\(^{50}\)
- haiku: example\(^{51}\), usage\(^{52}\)
- nature: example\(^{53}\), usage\(^{54}\)
- pyramid: example\(^{55}\), usage\(^{56}\)
- scrolls: example\(^{57}\), usage\(^{58}\)

\(^{40}\) https://www.sphinx-doc.org/en/master/theming.html#creating-themes
\(^{41}\) https://www.sphinx-doc.org/en/master/theming.html#builtin-themes
\(^{42}\) https://github.com/spatialaudio/nbsphinx/compare/agogo-theme...agogo-theme
\(^{43}\) https://nbsphinx.readthedocs.io/en/agogo-theme/
\(^{44}\) https://nbsphinx.readthedocs.io/en/alabaster-theme/
\(^{45}\) https://github.com/spatialaudio/nbsphinx/compare/alabaster-theme...alabaster-theme
\(^{46}\) https://nbsphinx.readthedocs.io/en/bizstyle-theme/
\(^{47}\) https://github.com/spatialaudio/nbsphinx/compare/bizstyle-theme...bizstyle-theme
\(^{48}\) https://nbsphinx.readthedocs.io/en/classic-theme/
\(^{49}\) https://github.com/spatialaudio/nbsphinx/compare/classic-theme...classic-theme
\(^{50}\) https://nbsphinx.readthedocs.io/en/haiku-theme/
\(^{51}\) https://github.com/spatialaudio/nbsphinx/compare/haiku-theme...haiku-theme
\(^{52}\) https://nbsphinx.readthedocs.io/en/nature-theme/
\(^{53}\) https://github.com/spatialaudio/nbsphinx/compare/nature-theme...nature-theme
\(^{54}\) https://nbsphinx.readthedocs.io/en/pyramid-theme/
\(^{55}\) https://github.com/spatialaudio/nbsphinx/compare/pyramid-theme...pyramid-theme
\(^{56}\) https://nbsphinx.readthedocs.io/en/scrolls-theme/
\(^{57}\) https://github.com/spatialaudio/nbsphinx/compare/scrolls-theme...scrolls-theme
• sphinxdoc: example\textsuperscript{59}, usage\textsuperscript{60}
• traditional: example\textsuperscript{61}, usage\textsuperscript{62}

2.5.2 3rd-Party Themes

• alabaster_jupyterhub\textsuperscript{63}: example\textsuperscript{64}, usage\textsuperscript{65}
• basicstrap\textsuperscript{66}: example\textsuperscript{67}, usage\textsuperscript{68}
• better\textsuperscript{69}: example\textsuperscript{70}, usage\textsuperscript{71}
• bootstrap\textsuperscript{72}: example\textsuperscript{73}, usage\textsuperscript{74}
• bootstrap-astropy\textsuperscript{75}: example\textsuperscript{76}, usage\textsuperscript{77}
• cloud/redcloud/greencloud\textsuperscript{78}: example\textsuperscript{79}, usage\textsuperscript{80}
• dask_sphinx_theme\textsuperscript{81}: example\textsuperscript{82}, usage\textsuperscript{83}
• guzzle_sphinx_theme\textsuperscript{84}: example\textsuperscript{85}, usage\textsuperscript{86}
• julia\textsuperscript{87}: example\textsuperscript{88}, usage\textsuperscript{89}
• jupyter\textsuperscript{90}: example\textsuperscript{91}, usage\textsuperscript{92}
• maisie_sphinx_theme\textsuperscript{93}: example\textsuperscript{94}, usage\textsuperscript{95}
• pandas_sphinx_theme\textsuperscript{96}: example\textsuperscript{97}, usage\textsuperscript{98}

\textsuperscript{59} https://nbsphinx.readthedocs.io/en/sphinxdoc-theme/
\textsuperscript{60} https://github.com/spatialaudio/nbsphinx/compare/sphinxdoc-theme...sphinxdoc-theme
\textsuperscript{61} https://nbsphinx.readthedocs.io/en/traditional-theme/
\textsuperscript{62} https://github.com/spatialaudio/nbsphinx/compare/traditional-theme...traditional-theme
\textsuperscript{63} https://github.com/jupyterhub/alabaster-jupyterhub
\textsuperscript{64} https://nbsphinx.readthedocs.io/en/alabaster-jupyterhub-theme/
\textsuperscript{65} https://github.com/spatialaudio/nbsphinx/compare/alabaster-jupyterhub-theme...alabaster-jupyterhub-theme
\textsuperscript{66} https://pythonhosted.org/sphinxjp.themes.basicstrap/
\textsuperscript{67} https://nbsphinx.readthedocs.io/en/basicstrap-theme/
\textsuperscript{68} https://github.com/spatialaudio/nbsphinx/compare/basicstrap-theme...basicstrap-theme
\textsuperscript{69} https://nbsphinx.readthedocs.io/en/better-theme/
\textsuperscript{70} https://github.com/spatialaudio/nbsphinx/compare/better-theme...better-theme
\textsuperscript{71} https://nbsphinx.readthedocs.io/en/bootstrap-theme/
\textsuperscript{72} https://github.com/spatialaudio/nbsphinx/compare/bootstrap-theme...bootstrap-theme
\textsuperscript{73} https://nbsphinx.readthedocs.io/en/astropy-
\textsuperscript{74} https://github.com/spatialaudio/nbsphinx/compare/astropy-astropy-theme
\textsuperscript{75} https://nbsphinx.readthedocs.io/en/cloud-
\textsuperscript{76} https://github.com/spatialaudio/nbsphinx/compare/cloud-theme...cloud-theme
\textsuperscript{77} https://nbsphinx.readthedocs.io/en/dask-
\textsuperscript{78} https://github.com/spatialaudio/nbsphinx/compare/dask-theme...dask-theme
\textsuperscript{79} https://nbsphinx.readthedocs.io/en/guzzle-
\textsuperscript{80} https://github.com/spatialaudio/nbsphinx/compare/guzzle-theme...guzzle-theme
\textsuperscript{81} https://nbsphinx.readthedocs.io/en/julia-
\textsuperscript{82} https://github.com/jupyter/jupyter-sphinx-theme/
\textsuperscript{83} https://nbsphinx.readthedocs.io/en/maisie-
\textsuperscript{84} https://github.com/pandas-dev/pandas-sphinx-theme
\textsuperscript{85} https://nbsphinx.readthedocs.io/en/pandas-
\textsuperscript{86} https://github.com/JuliaLang/JuliaDoc
\textsuperscript{87} https://nbsphinx.readthedocs.io/en/julia-
\textsuperscript{88} https://github.com/maisie-dev/maisie-sphinx-theme
\textsuperscript{89} https://nbsphinx.readthedocs.io/en/maisie-
\textsuperscript{90} https://github.com/spatialaudio/nbsphinx/compare/jupyter-theme...jupyter-theme
\textsuperscript{91} https://nbsphinx.readthedocs.io/en/jupyter-
\textsuperscript{92} https://github.com/spatialaudio/nbsphinx/compare/jupyter-theme...jupyter-theme
\textsuperscript{93} https://nbsphinx.readthedocs.io/en/maisie-
\textsuperscript{94} https://nbsphinx.readthedocs.io/en/jupyter-
\textsuperscript{95} https://github.com/spatialaudio/nbsphinx/compare/jupyter-theme...jupyter-theme
\textsuperscript{96} https://nbsphinx.readthedocs.io/en/pandas-
\textsuperscript{97} https://github.com/spatialaudio/nbsphinx/compare/pandas-theme...pandas-theme
2.6 Using Notebooks with Git

Git\textsuperscript{126} is extremely useful for managing source code and it can and should also be used for managing Jupyter notebooks. There is one caveat, however: Notebooks can contain output cells with rich media like images, plots, sounds, HTML, JavaScript and many other types of bulky machine-created content. This can make it hard to work with Git efficiently, because changes in those bulky contents can completely obscure the more interesting human-made changes in text and source code. Working with multiple collaborators on a notebook can become very tedious because of this.

It is therefore highly recommended that you remove all outputs from your notebooks before committing changes to a Git repository (except for the reasons mentioned in Pre-Executing Notebooks (page 39)).

If there are no output cells in a notebook, \texttt{nbsphinx} will by default execute the notebook, and the pages generated by Sphinx will therefore contain all the output cells. See Controlling Notebook Execution (page 39) for how this behavior can be customized.

\footnote{126} https://github.com/spatialaudio/nbsphinx/issues}

---

\textsuperscript{99} https://github.com/pangeo-data/sphinx_pangeo_theme/
\textsuperscript{100} https://nbsphinx.readthedocs.io/en/pangeo-theme/
\textsuperscript{101} https://github.com/spatialaudio/nbsphinx/compare/pangeo-theme%5E...pangeo-theme
\textsuperscript{102} https://schettino72.github.io/sphinx_press_site/
\textsuperscript{103} https://nbsphinx.readthedocs.io/en/press-theme/
\textsuperscript{104} https://github.com/spatialaudio/nbsphinx/compare/press-theme%5E...press-theme
\textsuperscript{105} https://github.com/shiftlab/pytorch_sphinx_theme
\textsuperscript{106} https://nbsphinx.readthedocs.io/en/pytorch-theme/
\textsuperscript{107} https://github.com/spatialaudio/nbsphinx/compare/pytorch-theme%5E...pytorch-theme
\textsuperscript{108} https://docs.red-dove.com/sphinx_sizzle_theme/
\textsuperscript{109} https://nbsphinx.readthedocs.io/en/sizzle-theme/
\textsuperscript{110} https://github.com/spatialaudio/nbsphinx/compare/sizzle-theme%5E...sizzle-theme
\textsuperscript{111} https://github.com/bashtage/sphinx-material
\textsuperscript{112} https://nbsphinx.readthedocs.io/en/material-theme/
\textsuperscript{113} https://github.com/spatialaudio/nbsphinx/compare/material-theme%5E...material-theme
\textsuperscript{114} https://github.com/ionelmc/sphinx-py3doc-enhanced-theme
\textsuperscript{115} https://nbsphinx.readthedocs.io/en/py3doc-enhanced-theme/
\textsuperscript{116} https://github.com/spatialaudio/nbsphinx/compare/py3doc-enhanced-theme%5E...py3doc-enhanced-theme
\textsuperscript{117} https://github.com/pyviz-dev/sphinx_pyviz_theme
\textsuperscript{118} https://github.com/spatialaudio/nbsphinx/compare/pyviz-theme%5E...pyviz-theme
\textsuperscript{119} https://github.com/spatialaudio/nbsphinx/compare/pyviz-theme%5E...pyviz-theme
\textsuperscript{120} https://github.com/readthedocs/sphinx_rtd_theme
\textsuperscript{121} https://nbsphinx.readthedocs.io/en/rtd-theme/
\textsuperscript{122} https://github.com/spatialaudio/nbsphinx/compare/rtd-theme%5E...rtd-theme
\textsuperscript{123} https://github.com/typlog/sphinx-typlog-theme
\textsuperscript{124} https://nbsphinx.readthedocs.io/en/typlog-theme/
\textsuperscript{125} https://github.com/spatialaudio/nbsphinx/compare/typlog-theme%5E...typlog-theme
\textsuperscript{126} https://github.com/spatialaudio/nbsphinx/issues
\textsuperscript{127} https://git-scm.com/
In the Jupyter Notebook application, you can manually clear all outputs by selecting “Cell” → “All Output” → “Clear” from the menu. In JupyterLab, the menu items are “Edit” → “Clear All Outputs”.

There are several tools available to remove outputs from multiple files at once without having to open them separately. You can even include such a tool as “clean/smudge filters” into your Git workflow, which will strip the output cells automatically whenever a Git command is executed. For details, have a look at those links:

- [https://github.com/kynan/nbstripout](https://github.com/kynan/nbstripout)
- [https://github.com/toobaz/ipynb_output_filter](https://github.com/toobaz/ipynb_output_filter)
- [https://pascalbugnion.net/blog/ipython-notebooks-and-git.html](https://pascalbugnion.net/blog/ipython-notebooks-and-git.html)
- [https://github.com/choldgraf/nbclean](https://github.com/choldgraf/nbclean)
- [https://jamesfolberth.org/articles/2017/08/07/git-commit-hook-for-jupyter-notebooks/](https://jamesfolberth.org/articles/2017/08/07/git-commit-hook-for-jupyter-notebooks/)

The following section was generated from doc/markdown-cells.ipynb

### 3 Markdown Cells

We can use *emphasis*, **boldface**, preformatted text.

It looks like strike-out text is not supported: [STRIKEOUT: strikethrough].

- Red
- Green
- Blue

1. One
2. Two
3. Three

Arbitrary Unicode characters should be supported, e.g. łśō. Note, however, that this only works if your HTML browser and your LaTeX processor provide the appropriate fonts.

#### 3.1 Equations

Inline equations like \(e^{i\pi} = -1\) can be created by putting a LaTeX expression between two dollar signs, like this: $\text{e}^{i\pi} = -1$.

Note

Avoid leading and trailing spaces around math expressions, otherwise errors like the following will occur when Sphinx is running:

```
ERROR: Unknown interpreted text role "raw-latex".
```

See also the pandoc docs\(^{128}\):

\(^{128}\) [https://pandoc.org/MANUAL.html#math](https://pandoc.org/MANUAL.html#math)
Equations can also be displayed on their own line like this:

\[ \int_{-\infty}^{\infty} f(x) \delta(x-x_0) \, dx = f(x_0). \]

This can be done by simply using one of the LaTeX math environments, like so:

```
\begin{equation}
\int_{-\infty}^{\infty} f(x) \delta(x-x_0) \, dx = f(x_0)
\end{equation}
```

Note

For equations to be shown in HTML output, you have to specify a math extension in your `extensions` setting, e.g.:

```
extensions = [
  'nbsphinx',
  'sphinx.ext.mathjax',
  # ... other useful extensions ...
]
```

### 3.1.1 Automatic Equation Numbering

This is not automatically enabled in Jupyter notebooks, but you can install a notebook extension in order to enable equation numbering: https://jupyter-contrib-nbextensions.readthedocs.io/en/latest/nbextensions/equation-numbering/readme.html.

Automatic Equation Numbering is enabled on https://nbviewer.jupyter.org/, see e.g. the latest version of this very notebook at the link https://nbviewer.jupyter.org/github/spatialaudio/nbsphinx/blob/master/doc/markdown-cells.ipynb#Automatic-Equation-Numbering.

When using nbSphinx, you can use the following `mathjax_config` setting in your `conf.py` file to enable automatic equation numbering in HTML output. In LaTeX output, the equations are numbered by default.

```
mathjax_config = {
  'TeX': {'equationNumbers': {'autoNumber': 'AMS', 'useLabelIds': True}},
}
```

You can use \texttt{\label{...}} to give a unique label to an equation:

\[ \phi = \frac{1 + \sqrt{5}}{2} \]

```
\begin{equation}
\phi = \frac{1 + \sqrt{5}}{2}
\label{golden-mean}
\end{equation}
```

If automatic equation numbering is enabled, you can later reference that equation using its label. You can use `\eqref{golden-mean}` for a reference with parentheses: (2), or `\ref{golden-mean}` for a reference without them: 2.

In HTML output, these equation references only work for equations within a single HTML page. In LaTeX output, equations from other notebooks can be referenced, e.g. (08.15).

### 3.1.2 Manual Equation Numbering

If you prefer to assign equation numbers (or some kind of names) manually, you can do so with `\tag{...}`:

\[
\begin{equation}
\tag{99.4}
\end{equation}
\]

The above equation has the number 99.4.

### 3.2 Citations


Example: [KRKP+16].

\[
\begin{cite}
data-cite="kluyver2016jupyter"
\end{cite}
\]

Kluyver et al. (2016)

You don’t actually have to use `<cite>`, any inline HTML tag can be used, e.g. `<strong>`: [PGH11].

\[
\begin{strong}
data-cite="perez2011python"
\end{strong}
\]

Python: An Ecosystem for Scientific Computing

You’ll also have to define a list of references, see the section about references (page 57).

There is also a Notebook extension which may or may not be useful: https://github.com/takluyver/cite2c.

### 3.3 Code

We can also write code with nice syntax highlighting:

```python
print("Hello, world!")
```

### 3.4 Tables

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>A and B</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>
3.5 Images

Local image:

![Jupyter notebook icon](images/notebook_icon.png)

Remote image:

![Python logo (remote)](https://www.python.org/static/img/python-logo-large.png)

3.5.1 SVG support for LaTeX

LaTeX doesn't support SVG images, but there are Sphinx extensions that can be used for automatically converting SVG images for inclusion in LaTeX output.

Just include one of the following options in the list of `extensions` (page 9) in your `conf.py` file.

- `'sphinxcontrib.inkscapeconverter'` or `'sphinxcontrib.rsvgconverter'`: See https://github.com/missinglinkelectronics/sphinxcontrib-svg2pdfconverter for installation instructions.

The external programs `inkscape` or `rsvg-convert` (Debian/Ubuntu package `librsvg2-bin`) are needed, respectively.

- `'sphinx.ext.imgconverter'`: This is a built-in Sphinx extension, see https://www.sphinx-doc.org/en/master/usage/extensions/imgconverter.html.

This needs the external program `convert` from `ImageMagick`.

The disadvantage of this extension is that SVGs are converted to bitmap images.

If one of those extensions is installed, SVG images can be used even for LaTeX output:
Remote SVG images can also be used (and will be shown in the LaTeX output):

This is a cell attachment:

In the Jupyter Notebook, there is a special “Attachments” cell toolbar which you can use to see all attachments of a cell and delete them, if needed.

### 3.7 HTML Elements (HTML only)

It is allowed to use plain HTML elements within Markdown cells. Those elements are passed through to the HTML output and are ignored for the LaTeX output. Below are a few examples.

HTML5 audio\(^{130}\) elements can be created like this:

Example:

The HTML audio element is not supported!

HTML5 video\(^{131}\) elements can be created like this:

\(^{130}\) https://developer.mozilla.org/en-US/docs/Web/HTML/Element/audio
Example:
The HTML video element is not supported!
The alternative text is shown in browsers that don’t support those elements. The same text is also shown in Sphinx’s LaTeX output.

**Note:** You can also use local files for the `<audio>` and `<video>` elements, but you have to create a link to the source file somewhere, because only then are the local files copied to the HTML output directory! You should do that anyway to make the audio/video file accessible to browsers that don’t support the `<audio>` and `<video>` elements.

### 3.8 Info/Warning Boxes

**Warning**
This is an experimental feature! Its usage will probably change in the future or it might be removed completely!

Until there is an info/warning extension for Markdown/CommonMark (see this issue[^132]), such boxes can be created by using HTML `<div>` elements like this:

```html
<div class="alert alert-info">
  Note
  This is a note!
</div>
```

For this to work reliably, you should obey the following guidelines:

- The `class` attribute has to be either "alert alert-info" or "alert alert-warning", other values will not be converted correctly.
- No further attributes are allowed.
- For compatibility with CommonMark, you should add an empty line between the `<div>` start tag and the beginning of the content.

**Note**
The text can contain further Markdown formatting. It is even possible to have nested boxes:

```markdown
... but please don’t overuse this!
```

[^132]: https://github.com/jupyter/notebook/issues/1292
3.9 Links to Other Notebooks

Relative links to local notebooks can be used: a link to a notebook in a subdirectory (page 47), a link to an orphan notebook (latter won’t work in LaTeX output, because orphan pages are not included there).

This is how a link is created in Markdown:

[a link to a notebook in a subdirectory](subdir/a-notebook-in-a-subdir.ipynb)

Markdown also supports reference-style links: a reference-style link (page 47), another version of the same link (page 47).

These can be created with this syntax:

[a reference-style link][mylink]

[mylink]: subdir/a-notebook-in-a-subdir.ipynb

Links to sub-sections are also possible, e.g. this subsection (page 48).

This link was created with:

[this subsection](subdir/a-notebook-in-a-subdir.ipynb#A-Sub-Section)

You just have to remember to replace spaces with hyphens!

BTW, links to sections of the current notebook work, too, e.g. beginning of this section (page 25).

This can be done, as expected, like this:

[beginning of this section](#Links-to-Other-Notebooks)

It’s also possible to create a link to the beginning of the current page (page ??), by simply using a # character:

[link to the beginning of the current page](#)

3.10 Links to *.rst Files (and Other Sphinx Source Files)

Links to files whose extension is in the configuration value source_suffix133, will be converted to links to the generated HTML/LaTeX pages. Example: A reStructuredText file (page 54).

This was created with:

[A reStructuredText file](a-normal-rst-file.rst)

Links to sub-sections are also possible. Example: Sphinx Directives (page 56).

This was created with:

[Sphinx Directives](a-normal-rst-file.rst#sphinx-directives-for-info-warning-boxes)

Note

Sphinx section anchors are different from Jupyter section anchors! To create a link to a subsection in an .rst file (or another non-notebook source file), you not only have to replace spaces with hyphens, but also slashes and some other characters. In case of doubt, just check the target HTML page generated by Sphinx.

---

133 https://www.sphinx-doc.org/en/master/config.html#confval-source_suffix
3.11 Links to Local Files

Links to local files (other than Jupyter notebooks and other Sphinx source files) are also possible, e.g. requirements.txt.

This was simply created with:

[requirements.txt](requirements.txt)

The linked files are automatically copied to the HTML output directory. For LaTeX output, links are created, but the files are not copied to the target directory.

3.12 Links to Domain Objects

Links to Sphinx domain objects\(^{134}\) (such as a Python class or JavaScript function) are also possible. For example: example_python_function() (page 56).

This was created with:

[example_python_function()](a-normal-rst-file.rst#example_python_function)

This is especially useful for use with the Sphinx autodoc\(^{135}\) extension!

---

The following section was generated from doc/code-cells.ipynb  

4 Code Cells

4.1 Code, Output, Streams

An empty code cell:

[ ]:

Two empty lines:

[ ]:

Leading/trailing empty lines:

[1]:

# 2 empty lines before, 1 after

A simple output:

[2]: 6 * 7

[2]: 42

The standard output stream:

[3]: print('Hello, world!')


Hello, world!

Normal output + standard output

[4]:

```
print('Hello, world!')
6 * 7
```

Hello, world!

[4]: 42

The standard error stream is highlighted and displayed just below the code cell. The standard output stream comes afterwards (with no special highlighting). Finally, the "normal" output is displayed.

[5]:

```
import sys

print("I'll appear on the standard error stream", file=sys.stderr)
print("I'll appear on the standard output stream")
"I'm the 'normal' output"
```

I'll appear on the standard output stream
I'll appear on the standard error stream

[5]: "I'm the 'normal' output"

---

Note

Using the IPython kernel, the order is actually mixed up, see https://github.com/ipython/ipykernel/issues/280.

---

### 4.2 Cell Magics

IPython can handle code in other languages by means of cell magics\(^\text{136}\):

[6]:

```
%%bash
for i in 1 2 3
do
  echo $i
done
```

```
1
2
3
```

---

### 4.3 Special Display Formats

See IPython example notebook\(^\text{137}\).

4.3.1 Local Image Files

```
from IPython.display import Image
i = Image(filename='images/notebook_icon.png')
i
```

```
from IPython.display import SVG
SVG(filename='images/python_logo.svg')
```

See also SVG support for LaTeX (page 22).

```
from IPython.display import Image
Image(url='https://www.python.org/static/img/python-logo-large.png')
```

```
from IPython.display import Image
Image(url='https://www.python.org/static/img/python-logo-large.png', embed=True)
```

4.3.2 Image URLs
4.3.3 Math

```python
from IPython.display import Math
eq = Math(r'\int_{-\infty}^{\infty} f(x) \delta(x - x_0) \, dx = f(x_0)')
eq
```

```latex
This is a \LaTeX{} equation: $a^2 + b^2 = c^2$
```

```latex
\begin{equation}
\int_{-\infty}^{\infty} f(x) \delta(x - x_0) \, dx = f(x_0)
\end{equation}
```

\[ \int_{-\infty}^{\infty} f(x) \delta(x - x_0) \, dx = f(x_0) \quad (3) \]
4.3.4 Plots

The output formats for Matplotlib plots can be customized. You’ll need separate settings for the Jupyter Notebook application and for nbsphinx.

If you want to use SVG images for Matplotlib plots, add this line to your IPython configuration file:

```
c.InlineBackend.figure_formats = {'svg'}
```

If you want SVG images, but also want nice plots when exporting to LaTeX/PDF, you can select:

```
c.InlineBackend.figure_formats = {'svg', 'pdf'}
```

If you want to use the default PNG plots or HiDPI plots using 'png2x' (a.k.a. 'retina'), make sure to set this:

```
c.InlineBackend.rc = {'figure.dpi': 96}
```

This is needed because the default 'figure.dpi' value of 72 is only valid for the Qt Console\textsuperscript{138}.

If you are planning to store your SVG plots as part of your notebooks, you should also have a look at the 'svg.hashsalt' setting.

For more details on these and other settings, have a look at Default Values for Matplotlib’s “inline” Backend\textsuperscript{139}.

The configuration file `ipython_kernel_config.py` can be either in the directory where your notebook is located (see the `ipython_kernel_config.py` in this directory), or in your profile directory (typically `~/.ipython/profile_default/ipython_kernel_config.py`). To find out your IPython profile directory, use this command:

```
python3 -m IPython profile locate
```

A local `ipython_kernel_config.py` in the notebook directory also works on https://mybinder.org/.

Alternatively, you can create a file with those settings in a file named `.ipython/profile_default/ipython_kernel_config.py` in your repository.

To get SVG and PDF plots for nbsphinx, use something like this in your `conf.py` file:

```
nbsphinx_execute_arguments = [  
    "--InlineBackend.figure_formats={'svg', 'pdf'"],
    "--InlineBackend.rc={'figure.dpi': 96}"],
]
```

In the following example, nbsphinx should use an SVG image in the HTML output and a PDF image for LaTeX/PDF output.

```
[17]: import matplotlib.pyplot as plt

[18]: fig, ax = plt.subplots(figsize=[6, 3])
    ax.plot([4, 9, 7, 20, 6, 33, 13, 23, 16, 62, 8]);
```

\textsuperscript{138} https://qtconsole.readthedocs.io/
\textsuperscript{139} https://nbviewer.jupyter.org/github/mgeier/python-audio/blob/master/plotting/matplotlib-inline-defaults.ipynb
Alternatively, the figure format(s) can also be chosen directly in the notebook (which overrides the setting in `nbsphinx_execute_arguments` and in the IPython configuration):

```python
[19]: %config InlineBackend.figure_formats = ['png']

[20]: fig

If you want to use PNG images, but with HiDPI resolution, use the special `'png2x'` (a.k.a. `'retina'`) format (which also looks nice in the LaTeX output):

```python
[21]: %config InlineBackend.figure_formats = ['png2x']

[22]: fig
4.3.5 Pandas Dataframes

Pandas dataframes\(^\text{140}\) should be displayed as nicely formatted HTML tables (if you are using HTML output).

```python
[23]: import numpy as np
import pandas as pd

[24]: df = pd.DataFrame(np.random.randint(0, 100, size=[5, 4]),
                      columns=['a', 'b', 'c', 'd'])

[24]:
   a  b  c  d
0  21 13 21 56
1  87 30 80 59
2  11 85 79 73
3  89 35 29 46
4  93 76 55 45
```

For LaTeX output, however, the plain text output is used by default. To get nice LaTeX tables, a few settings have to be changed:

```python
[25]: pd.set_option('display.latex.repr', True)

This is not enabled by default because of Pandas issue \#12182\(^\text{141}\).

The generated LaTeX tables utilize the `booktabs` package, so you have to make sure that package is loaded in the preamble\(^\text{142}\) with:

```
\usepackage{booktabs}
```

In order to allow page breaks within tables, you should use:

```python
[26]: pd.set_option('display.latex.longtable', True)
```

The `longtable` package is already used by Sphinx, so you don’t have to manually load it in the preamble.

\(^\text{140}\) https://pandas.pydata.org/pandas-docs/stable/getting_started/dsintro.html#dataframe
\(^\text{141}\) https://github.com/pandas-dev/pandas/issues/12182
\(^\text{142}\) https://www.sphinx-doc.org/en/master/latex.html
Finally, if you want to use LaTeX math expressions in your dataframe, you’ll have to disable escaping:

```
[27]: pd.set_option('display.latex.escape', False)
```

The above settings should have no influence on the HTML output, but the LaTeX output should now look nicer:

```
[28]: df = pd.DataFrame(np.random.randint(0, 100, size=[10, 4]),
                     columns=[r'$\alpha$', r'$\beta$', r'$\gamma$', r'$\delta$'])
```

```
<table>
<thead>
<tr>
<th></th>
<th>(\alpha)</th>
<th>(\beta)</th>
<th>(\gamma)</th>
<th>(\delta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>26</td>
<td>88</td>
<td>0</td>
<td>64</td>
</tr>
<tr>
<td>1</td>
<td>99</td>
<td>16</td>
<td>48</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>72</td>
<td>95</td>
<td>58</td>
<td>64</td>
</tr>
<tr>
<td>3</td>
<td>81</td>
<td>61</td>
<td>47</td>
<td>69</td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>70</td>
<td>14</td>
<td>57</td>
</tr>
<tr>
<td>5</td>
<td>27</td>
<td>59</td>
<td>38</td>
<td>90</td>
</tr>
<tr>
<td>6</td>
<td>54</td>
<td>3</td>
<td>45</td>
<td>82</td>
</tr>
<tr>
<td>7</td>
<td>48</td>
<td>17</td>
<td>87</td>
<td>90</td>
</tr>
<tr>
<td>8</td>
<td>74</td>
<td>86</td>
<td>71</td>
<td>89</td>
</tr>
<tr>
<td>9</td>
<td>48</td>
<td>51</td>
<td>20</td>
<td>72</td>
</tr>
</tbody>
</table>
```

### 4.3.6 YouTube Videos

```
[29]: from IPython.display import YouTubeVideo
```

YouTubeVideo('WAikxUGbomY')

![YouTube Video](https://i.imgur.com/WAikxUGbomY)
4.3.7 Interactive Widgets (HTML only)

The basic widget infrastructure is provided by the ipywidgets module. More advanced widgets are available in separate packages, see for example https://jupyter.org/widgets.

The JavaScript code which is needed to display Jupyter widgets is loaded automatically (using RequireJS). If you want to use non-default URLs or local files, you can use the `nbsphinx_widgets_path` (page 13) and `nbsphinx_requirejs_path` (page 12) settings.

```python
[30]: import ipywidgets as w
```

```python
[31]: slider = w.IntSlider()
    slider.value = 42
    slider
```

```
IntSlider(value=42)
```

A widget typically consists of a so-called “model” and a “view” into that model.

If you display a widget multiple times, all instances act as a “view” into the same “model”. That means that their state is synchronized. You can move either one of these sliders to try this out:

```python
[32]: slider
```

```
IntSlider(value=42)
```

You can also link different widgets.

Widgets can be linked via the kernel (which of course only works while a kernel is running) or directly in the client (which even works in the rendered HTML pages).

Widgets can be linked uni- or bi-directionally.

Examples for all 4 combinations are shown here:

```python
[33]: link = w.IntSlider(description='link')
    w.link((slider, 'value'), (link, 'value'))
    jslink = w.IntSlider(description='jslink')
    w.jslink((slider, 'value'), (jslink, 'value'))
    dlink = w.IntSlider(description='dlink')
    w.dlink((slider, 'value'), (dlink, 'value'))
    jsdlink = w.IntSlider(description='jsdlink')
    w.jsdlink((slider, 'value'), (jsdlink, 'value'))
    VBox([link, jslink, dlink, jsdlink])
```

```
VBox(children=(IntSlider(value=42, description='link'), IntSlider(value=0, description='jslink'), IntSlider(value=0, description='dlink'), IntSlider(value=0, description='jsdlink')))
```

Other Languages

The examples shown here are using Python, but the widget technology can also be used with different Jupyter kernels (i.e. with different programming languages).

[143] https://ipywidgets.readthedocs.io/
4.3.8 Arbitrary JavaScript Output (HTML only)

```javascript
var text = document.createTextNode("Hello, I was generated with JavaScript!");
// Content appended to "element" will be visible in the output area:
element.appendChild(text);
```

4.3.9 Unsupported Output Types

If a code cell produces data with an unsupported MIME type, the Jupyter Notebook doesn’t generate any output. nbSphinx, however, shows a warning message.

```python
display({
    'text/x-python': 'print("Hello, world!"),
    'text/x-haskell': 'main = putStrLn "Hello, world!"
}, raw=True)
```

Data type cannot be displayed: text/x-python, text/x-haskell

4.4 ANSI Colors

The standard output and standard error streams may contain ANSI escape sequences\(^\text{144}\) to change the text and background colors.

```python
print('BEWARE: \x1b[1;33;41mugly colors\x1b[0m', file=sys.stderr)
print('AB\x1b[43mCD\x1b[35mEF\x1b[1mGH\x1b[4mIJ\x1b[7mK','
'KL\x1b[49mMN\x1b[39mOP\x1b[22mQR\x1b[24mST\x1b[27mUV')
```

<table>
<thead>
<tr>
<th>ABCD</th>
<th>EFGH</th>
<th>IJKL</th>
<th>MNOP</th>
<th>QRSTUV</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEWARE:</td>
<td>ugly colors!</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The following code showing the 8 basic ANSI colors is based on [http://tldp.org/HOWTO/Bash-Prompt-HOWTO/x329.html](http://tldp.org/HOWTO/Bash-Prompt-HOWTO/x329.html). Each of the 8 colors has an “intense” variation, which is used for bold text.

```python
text = ' XYZ'
formatstring = '\x1b[{}m' + text + '\x1b[0m'

for fg in range(30, 38):
    for bg in range(40, 48):
        for bold in False, True:
            fg_code = ('1;' if bold else '') + str(fg)
            print('{}{:>4}'.format(fg_code) + formatstring.format(fg_code) +
                  '{}'.format(bg) + ':' + str(bg))
```

<table>
<thead>
<tr>
<th>30</th>
<th>40</th>
<th>41</th>
<th>42</th>
<th>43</th>
<th>44</th>
<th>45</th>
<th>46</th>
<th>47</th>
</tr>
</thead>
<tbody>
<tr>
<td>XYZ</td>
<td>XYZ</td>
<td>XYZ</td>
<td>XYZ</td>
<td>XYZ</td>
<td>XYZ</td>
<td>XYZ</td>
<td>XYZ</td>
<td>XYZ</td>
</tr>
<tr>
<td>1;30</td>
<td>XYZ</td>
<td>XYZ</td>
<td>XYZ</td>
<td>XYZ</td>
<td>XYZ</td>
<td>XYZ</td>
<td>XYZ</td>
<td>XYZ</td>
</tr>
</tbody>
</table>

ANSI also supports a set of 256 indexed colors. The following code showing all of them is based on http://bitmote.com/index.php?post/2012/11/19/Using-ANSI-Color-Codes-to-Colorize-Your-Bash-Prompt-on-Linux:

```
[38]: formatstring = '{\x1b[38;5;{0};48;5;{0}mX\x1b[1mX\x1b[m'

print(' + ' + '.join('{{:2}'.format(i) for i in range(36)))
print(' 0 ' + '.join(formatstring.format(i) for i in range(16)))
for i in range(7):
    i = i * 36 + 16
    print('{{:3} '.format(i) + '.join(formatstring.format(i + j)
            for j in range(36) if i + j < 256))
```

You can even use 24-bit RGB colors:

```
[39]: start = 255, 0, 0
end = 0, 0, 255
length = 79
out = []

for i in range(length):
    rgb = [start[c] + int((end[c] - start[c]) / length) for c in range(3)]
    out.append('\x1b['
        '38;2;{{rgb[2]}};{{rgb[1]}};{{rgb[0]}};
        '48;2;{{rgb[0]}};{{rgb[1]}};{{rgb[2]}}mX\x1b[m'.format(rgba=rgb))
print(''.join(out))
```

---

5 Raw Cells

The “Raw NBConvert” cell type can be used to render different code formats into HTML or LaTeX by Sphinx. This information is stored in the notebook metadata and converted appropriately.

5.1 Usage

To select a desired format from within Jupyter, select the cell containing your special code and choose options from the following dropdown menus:

1. Select “Raw NBConvert”
2. Switch the Cell Toolbar to “Raw Cell Format”
3. Chose the appropriate “Raw NBConvert Format” within the cell

5.2 Available Raw Cell Formats

The following examples show how different Jupyter cell formats are rendered by Sphinx.
5.2.1 None

By default (if no cell format is selected), the cell content is included (without any conversion) in both the HTML and LaTeX output. This is typically not useful at all.

"I'm a raw cell with no format."

5.2.2 reST

Raw cells in “reST” format are interpreted as reStructuredText and parsed by Sphinx. The result is visible in both HTML and LaTeX output.

“I'm a raw cell in reST\(^{146}\) format.”

5.2.3 Markdown

Raw cells in “Markdown” format are interpreted as Markdown, and the result is included in both HTML and LaTeX output. Since the Jupyter Notebook also supports normal Markdown cells, this might not be useful at all.

“I'm a raw cell in Markdown\(^{147}\) format.”

5.2.4 HTML

Raw cells in “HTML” format are only visible in HTML output. This option might not be very useful, since raw HTML code is also allowed within normal Markdown cells.

5.2.5 LaTeX

Raw cells in “LaTeX” format are only visible in LaTeX output.

I'm a raw cell in \LaTeX\(^{148}\) format.

5.2.6 Python

Raw cells in “Python” format are not visible at all (nor executed in any way).

The following section was generated from doc/hidden-cells.ipynb

6 Hidden Cells

You can remove cells from the HTML/LaTeX output by adding this to the cell metadata:

"nbsphinx": "hidden"

Hidden cells are still executed but removed afterwards.

For example, the following hidden cell defines the variable answer.

This is the cell after the hidden cell. Although the previous cell is not visible, its result is still available:

```
[2] : answer
```

\(^{146}\) https://www.sphinx-doc.org/rest.html

\(^{147}\) https://daringfireball.net/projects/markdown/

38
Don’t overuse this, because it may make it harder to follow what’s going on in your notebook. Also Markdown cells can be hidden. The following cell is hidden.

This is the cell after the hidden cell.

The following section was generated from doc/hidden-cells.ipynb

7 Controlling Notebook Execution

Notebooks with no outputs are automatically executed during the Sphinx build process. If, however, there is at least one output cell present, the notebook is not evaluated and included as is.

The following notebooks show how this default behavior can be used and customized.

7.1 Pre-Executing Notebooks

Automatically executing notebooks during the Sphinx build process is an important feature of nbsphinx. However, there are a few use cases where pre-executing a notebook and storing the outputs might be preferable. Storing any output will, by default, stop nbsphinx from executing the notebook.

7.1.1 Long-Running Cells

If you are doing some very time-consuming computations, it might not be feasible to re-execute the notebook every time you build your Sphinx documentation.

So just do it once – when you happen to have the time – and then just keep the output.

```python
[1]: import time

[2]: %time time.sleep(60 * 60)
   6 * 7
CPU times: user 160 ms, sys: 56 ms, total: 216 ms
Wall time: 1h 1s

[2]: 42
```

7.1.2 Rare Libraries

You might have created results with a library that’s hard to install and therefore you have only managed to install it on one very old computer in the basement, so you probably cannot run this whenever you build your Sphinx docs.

```python
[3]: from a_very_rare_library import calculate_the_answer

[4]: calculate_the_answer()
[4]: 42
```
7.1.3 Exceptions

If an exception is raised during the Sphinx build process, it is stopped (the build process, not the exception!). If you want to show to your audience how an exception looks like, you have two choices:

1. Allow errors – either generally or on a per-notebook or per-cell basis – see Ignoring Errors (page 41) (per cell (page 42)).
2. Execute the notebook beforehand and save the results, like it’s done in this example notebook:

```bash
[5]: 1 / 0
---------------------------------------------------------------------------
ZeroDivisionError Traceback (most recent call last)
<ipython-input-5-b710d87c980c> in <module>()
----> 1 1 / 0
ZeroDivisionError: division by zero
```

7.1.4 Client-specific Outputs

When nbsphinx executes notebooks, it uses the nbconvert module to do so. Certain Jupyter clients might produce output that differs from what nbconvert would produce. To preserve those original outputs, the notebook has to be executed and saved before running Sphinx.

For example, the JupyterLab help system shows the help text as cell outputs, while executing with nbconvert doesn’t produce any output.

```bash
[6]: sorted?
Signature: sorted(iterable, /, *, key=None, reverse=False)
Docstring:
Return a new list containing all items from the iterable in ascending order.
A custom key function can be supplied to customize the sort order, and the reverse flag can be set to request the result in descending order.
Type:   builtin_function_or_method
```

The following section was generated from doc/never-execute.ipynb

7.2 Explicitly Dis-/Enabling Notebook Execution

If you want to include a notebook without outputs and yet don’t want nbsphinx to execute it for you, you can explicitly disable this feature.

You can do this globally by setting the following option in conf.py:

```bash
nbsphinx_execute = 'never'
```

Or on a per-notebook basis by adding this to the notebook’s JSON metadata:

```bash
"nbsphinx": {
  "execute": "never"
},
```

There are three possible settings, "always", "auto" and "never". By default (= "auto"), notebooks with no outputs are executed and notebooks with at least one output are not. As always, per-notebook settings take precedence over the settings in conf.py.
This very notebook has its metadata set to "never", therefore the following cell is not executed:

```
[ ]: 6 * 7
```

The following section was generated from `doc/allow-errors.ipynb`

### 7.3 Ignoring Errors

Normally, if an exception is raised while executing a notebook, the Sphinx build process is stopped immediately.

If a notebook contains errors on purpose (or if you are too lazy to fix them right now), you have four options:

1. Manually execute the notebook in question and save the results, see the pre-executed example notebook (page 39).
2. Allow errors in all notebooks by setting this option in `conf.py`: `nbsphinx_allow_errors = True`
3. Allow errors on a per-notebook basis by adding this to the notebook’s JSON metadata: "nbsphinx": { "allow_errors": true },
4. Allow errors on a per-cell basis using the `raises-exception` tag, see Ignoring Errors on a Cell-by-Cell Basis (page 42).

This very notebook is an example for the third option. The results of the following code cells are not stored within the notebook, therefore it is executed during the Sphinx build process. Since the above-mentioned `allow_errors` flag is set in this notebook’s metadata, all cells are executed although most of them cause an exception.

```
[1]: nonsense
```

```
NameError Traceback (most recent call last)
<ipython-input-1-7dd4c0df649c> in <module>
      1 nonsense
----> 2
NameError: name 'nonsense' is not defined
```

```
[2]: 42 / 0
```

```
ZeroDivisionError Traceback (most recent call last)
<ipython-input-2-52cebea8b64f> in <module>
      1 42 / 0
----> 2
ZeroDivisionError: division by zero
```

```
[3]: print 'Hello, world!'
```

```
File "<ipython-input-3-653b30cd70a8>", line 1
  print 'Hello, world!'
SyntaxError: Missing parentheses in call to 'print'. Did you mean print('Hello, world!')?
```

```
[4]: 6 - 7
```

41
7.4 Ignoring Errors on a Per-Cell Basis

Instead of ignoring errors for all notebooks or for some selected notebooks (see the previous notebook (page 41)), you can be more fine-grained and just allow errors on certain code cells by tagging them with the raises-exception tag.

The following code cell has the raises-exception tag.

```python
problem
```

```
NameError: name 'problem' is not defined
```

The following code cell is executed even though the previous cell raised an exception.

```python
'no problem'
```

Note

The behavior of the raises-exception tag doesn’t match its name. While it does allow exceptions, it does not check if an exception is actually raised!

This will hopefully be fixed at some point, see https://github.com/jupyter/nbconvert/issues/730.

7.5 Configuring the Kernels

7.5.1 Kernel Name

If we have multiple kernels installed, we can choose to override the kernel saved in the notebook using nbsphinx_kernel_name (page 11):
which uses the kernel named `python-upstream-dev` instead of the kernel name stored in the notebook.

### 7.5.2 Kernel Arguments

We can also pass options to the kernel by setting `nbsphinx_execute_arguments` (page 11) in `conf.py`. These work the same way as `ipython_kernel_config.py`. For example, using

```python
nbsphinx_execute_arguments = [
    "--InlineBackend.rc={'figure.dpi': 96},"
]
```

to set plot options (page 30) is the same as writing:

```python
c.InlineBackend.rc = {'figure.dpi': 96}
```

in `ipython_kernel_config.py` or using:

```
%config InlineBackend.rc={'figure.dpi': 96}
```

at the top of a notebook:

```
[1]: get_ipython().config.InlineBackend.rc
[1]: {'figure.dpi': 96}
```

### 7.5.3 Environment Variables

The contents of `os.environ` after the execution of `conf.py` will be passed as environment variables to the kernel. As an example, `MY_DUMMY_VARIABLE` has been set in `conf.py` like this:

```python
import os
os.environ['MY_DUMMY_VARIABLE'] = 'Hello from conf.py!
```

... and it can be checked in the notebook like this:

```
[2]: import os
[2]: os.environ['MY_DUMMY_VARIABLE']
[2]: 'Hello from conf.py!
```

This is useful if we want to edit `PYTHONPATH` in order to compile the documentation without installing the project:

```python
import os
src = os.path.abspath('../src')
os.environ['PYTHONPATH'] = src
```

If you are using `https://mybinder.org/` and you want to define environment variables, you should create a file `.binder/start` in your repository (see Binder docs\[146\]) containing definitions like this:

\[146\] https://mybinder.readthedocs.io/en/latest/config_files.html#start-run-code-before-the-user-sessions-starts
7.6 Cell Execution Timeout

By default, code cells will be executed until they are finished, even if that takes a very long time. In some cases they might never finish.

If you would like to only use a finite amount of time per cell, you can choose a timeout length for all notebooks by setting the following option in `conf.py`:

```python
nbsphinx_timeout = 60
```

Or change the timeout length on a per-notebook basis by adding this to the notebook’s JSON metadata:

```
"nbsphinx": {
    "timeout": 60
},
```

The timeout is given in seconds, use `-1` to disable the timeout (which is the default).

Alternatively, you can manually execute the notebook in question and save the results, see the pre-executed example notebook (page 39).

8 Prolog and Epilog

When including notebooks in your Sphinx documentation, you can choose to add some generic content before and after each notebook. This can be done with the configuration values `nbsphinx_prolog` and `nbsphinx_epilog` in the file `conf.py`.

The prolog and epilog strings can hold arbitrary reST\(^\text{149}\) markup. Particularly, the `only\(^\text{150}\)` and `raw\(^\text{151}\)` directives can be used to have different content for HTML and LaTeX output.

Those strings are also processed by the Jinja\(^\text{152}\) templating engine. This means you can run Python-like code within those strings. You have access to the current Sphinx build environment\(^\text{153}\) via the variable `env`. Most notably, you can get the file name of the current notebook with

```python
{{ env.doc2path(env.docname, base=None) }}
```

Have a look at the Jinja2 template documentation\(^\text{154}\) for more information.

---

\(^{149}\) [https://www.sphinx-doc.org/rest.html](https://www.sphinx-doc.org/rest.html)


\(^{152}\) [http://jinja.pocoo.org/](http://jinja.pocoo.org/)


\(^{154}\) [http://jinja.pocoo.org/docs/latest/templates/](http://jinja.pocoo.org/docs/latest/templates/)
This is especially prone to happen when using raw LaTeX, with its abundance of braces. To avoid clashing braces you can try to insert additional spaces or LaTeX macros that don’t have a visible effect, like e.g. \strut{}. For example, you can avoid three consecutive opening braces with something like that:
\texttt{\strut{env.doc2path(env.docname, base=None)}}</tt>

NB: The three consecutive closing braces in this example are not problematic.

An alternative work-around would be to surround LaTeX braces with Jinja braces like this:
{{ '{' }}

The string within will not be touched by Jinja.

Another special Jinja syntax is {{ %}}, which is also often used in fancy TeX/LaTeX code. A work-around for this situation would be to use
{{ '{' %'}}

### 8.1 Examples

You can include a simple static string, using reST\textsuperscript{155} markup if you like:

```python
nbsphinx_epilog = ""
```

Generated by nbsphinx\_ from a Jupyter\_ notebook.

```makefile
.. _nbsphinx: https://nbsphinx.readthedocs.io/
.. _Jupyter: https://jupyter.org/
```

Using some additional Jinja\_2 markup and the information from the env variable, you can create URLs that point to the current notebook file, but located on some other server:

```python
nbsphinx_prolog = ""
Go there: https://example.org/notebooks/{{ env.doc2path(env.docname, base=None) }}
```

You can also use separate content for HTML and LaTeX output, e.g.:

```python
nbsphinx_prolog = r""
{% set docname = env.doc2path(env.docname, base=None) %}
```

```makefile
.. only:: html
    Go there: https://example.org/notebooks/{{ docname }}

.. raw:: latex

    \nbsphinxstartnotebook{The following section was created from \texttt{\strut{docname}}}:
```

\textsuperscript{155} https://www.sphinx-doc.org/rest.html
Note the use of the `\nbsphinxstartnotebook` and `\nbsphinxstopnotebook` commands. Those make sure there is not too much space between the “prolog” and the beginning of the notebook and, respectively, between the end of the notebook and the “epilog”. They also avoid page breaks, in order for the “prolog”/“epilog” not to end up on the page before/after the notebook.

For a more involved example for different HTML and LaTeX versions, see the file `conf.py` of the nbsphinx documentation.

The following section was generated from `doc/custom-formats.pct.py` ...

## 9 Custom Notebook Formats

By default, Jupyter notebooks are stored in files with the suffix `.ipynb`, which use the JSON format for storage.

However, there are libraries available which allow storing notebooks in different formats, using different file suffixes.

To use a custom notebook format in nbsphinx, you can specify the `nbsphinx_custom_formats` option in your `conf.py` file. You have to provide the file extension and a conversion function that takes the contents of a file (as a string) and returns a Jupyter notebook object.

```python
nbsphinx_custom_formats = {
    '.mysuffix': 'mylibrary.converter_function',
}
```

The converter function can be given as a string (recommended) or as a function object.

If a conversion function takes more than a single string argument, you can specify the function name plus a dictionary with keyword arguments which will be passed to the conversion function in addition to the file contents.

```python
nbsphinx_custom_formats = {
    '.mysuffix': ['mylibrary.converter_function', {'some_arg': 42}],
}
```

You can of course use multiple formats by specifying multiple conversion functions.

### 9.1 Example: Jupyter

One example for a library which provides a custom conversion function is [jupyter][1], which allows storing the contents of Jupyter notebooks in Markdown and R-Markdown, as well as plain Julia, Python and R files.

Since its conversion function takes more than a single string argument, we have to pass a keyword argument, e.g.:

---

[1]: https://github.com/mwouts/jupyter
This very page is an example of a notebook stored in the `py:percent` format (see docs\textsuperscript{157}):

```python
[nbsphinx_custom_formats = {
    '.Rmd': ['jupyter.text.reads', {'fmt': 'Rmd'}],
}
```

To select a suitable conversion function, we use the following setting in `conf.py`:

```python
[nbsphinx_custom_formats = {
    '.pct.py': ['jupyter.text.reads', {'fmt': 'py:percent'}],
}
```

Another example is this gallery example page (page 57).

---

10 **Notebooks in Sub-Directories**

You can organize your notebooks in subdirectories and `nbsphinx` will take care that relative links to other notebooks, images and other files still work.

Let's see if links to local images work:

```python
[1]: from IPython.display import Image
Image(filename='..//images/notebook_icon.png')
```

\textsuperscript{157} https://jupyter.readthedocs.io/en/latest/formats.html#the-percent-format
Warning
There may be problems with images in output cells if your source directory contains symbolic links, see issue #49158.

A link to a notebook in the same sub-directory: \textit{link} (page 53).
A link to a notebook in the parent directory: \textit{link} (page 19).
A link to a local file: \textit{link}.
A random equation:

\[ F_n = F_{n-1} + F_{n-2} \]  \hspace{1cm} (08.15)

10.1 A Sub-Section

This is just for testing inter-notebook links, see \textit{this section} (page 25).

The following section was generated from doc/subdir/gallery.ipynb

11 Creating Thumbnail Galleries

Inspired by \textit{Sphinx-Gallery}159, you can create thumbnail galleries from a list of Jupyter notebooks (or other Sphinx source files).

\textit{nbsphinx} does not provide any gallery styles, but you can easily use the styles from Sphinx-Gallery by installing it:

\texttt{python3 \textendash;m pip install sphinx-gallery \textendash;user}

\ldots and loading the styles in your \texttt{conf.py} with:

\begin{verbatim}
extensions = [
    'nbsphinx',
    'sphinx_gallery.load_style',
    # more extensions, if needed ...
]
\end{verbatim}

You’ll need Sphinx-Gallery version 0.6 or higher.

However, you can also create your own CSS styles if you prefer (then you don’t need to install Sphinx-Gallery). You can load your CSS files with \texttt{html_css_files160}.

\begin{itemize}
    \item \texttt{https://github.com/spatialaudio/nbsphinx/issues/49}
    \item \texttt{https://sphinx-gallery.github.io/}
    \item \texttt{https://www.sphinx-doc.org/en/master/usage/configuration.html\#confval-html_css_files}
\end{itemize}

158 https://github.com/spatialaudio/nbsphinx/issues/49
159 https://sphinx-gallery.github.io/
You can create *Thumbnail Galleries in reST Files* (page 57) and you can create galleries by adding the "nbsphinx-gallery" cell tag or metadata to notebooks, which is used just like the "nbsphinx-toctree" (page 53) cell tag/metadata.

For possible options, see the *toctree* (page 53) notebook.

Note

In LaTeX output this behaves just like toctree, i.e. no thumbnail gallery is shown, but the linked files are included in the document.

Like with toctree you should avoid adding content after a gallery (except other toctrees and galleries) because this content would appear in the LaTeX output after the content of all included source files, which is probably not what you want.

The following cell has the "nbsphinx-gallery" tag, which creates a thumbnail gallery. The *first* section title in that cell (if available) is used as "caption" (unless it’s given in the metadata).

The notebooks in the following gallery describe different ways how to select which images are used as thumbnails.

The following section was generated from doc/gallery/cell-tag.ipynb ...

11.1 Using a Cell Tag to Select a Thumbnail

You can select any code cell (with appropriate output) by tagging it with the nbsphinx-thumbnail tag.

If there are multiple outputs in the selected cell, the last one is used. See *Choosing from Multiple Outputs* (page 51) for how to select a specific output. If you want to show a tooltip, have a look at *Using Cell Metadata to Select a Thumbnail* (page 50).

[1]: `import matplotlib.pyplot as plt`

The following cell has the nbsphinx-thumbnail tag:

[2]: `fig, ax = plt.subplots(figsize=[6, 3])
ax.plot([4, 9, 7, 20, 6, 33, 13, 23, 16, 62, 8])`

[2]: `[<matplotlib.lines.Line2D at 0x7fa140a76748>]

![Graph]

---

doc/gallery/cell-tag.ipynb ends here.
11.2 Using Cell Metadata to Select a Thumbnail

If the \texttt{nbsphinx-thumbnail} (page 49) cell tag is not enough, you can use cell metadata to specify more options.

The last cell in this notebook has this metadata:

```json
{
    "nbsphinx-thumbnail": {
        "tooltip": "This tooltip message was defined in cell metadata"
    }
}
```

If there are multiple outputs in the selected cell, the last one is used. See \textit{Choosing from Multiple Outputs} (page 51) for how to select a specific output.

```python
[1]: import matplotlib.pyplot as plt
import numpy as np

[2]: plt.rcParams['image.cmap'] = 'coolwarm'
plt.rcParams['image.origin'] = 'lower'

Some example data stolen from https://matplotlib.org/examples/pylab_examples/pcolor_demo.html:

[3]: x, y = np.meshgrid(np.arange(-3, 3, 0.1), np.arange(-2, 2, 0.1))
z = (1 - x / 2 + x ** 5 + y ** 3) * np.exp(-x ** 2 - y ** 2)

[4]: zmax = np.max(np.abs(z))

[5]: fig, ax = plt.subplots(figsize=[5, 3.5])
ax.imshow(z, vmin=-zmax, vmax=zmax)

[5]: <matplotlib.image.AxesImage at 0x7f198930c630>
```

![Image of the generated output](doc/gallery/cell-metadata.ipynb)
11.3 Choosing from Multiple Outputs

By default, the last output of the selected cell is used as a thumbnail. If that’s what you want, you can simply use the `nbsphinx-thumbnail` (page 49) cell tag.

If you want to specify one of multiple outputs, you can add a (zero-based) "output-index" to your "nbsphinx-thumbnail" cell metadata.

The following cell has this metadata, selecting the third output to be used as thumbnail in the gallery (page 48).

```json
{
    "nbsphinx-thumbnail": {
        "output-index": 2
    }
}
```

[1]: `from IPython.display import Image`
```
display(Image(url='https://jupyter.org/assets/nav_logo.svg'))
print('Hello!')
display(Image(filename='../images/notebook_icon.png'))
display(Image(url='https://www.python.org/static/img/python-logo-large.png', embed=True))
```

Hello!
11.4 A Notebook without Thumbnail

This notebook doesn’t contain any thumbnail metadata. It should be displayed with the default thumbnail image in the gallery (page 48).

11.5 Specifying Thumbnails in conf.py

This notebook doesn’t contain any thumbnail metadata. But in the file conf.py, a thumbnail is specified (via the nbsphinx_thumbnails (page 12) option), which will be used in the gallery (page 48).

The keys in the nbsphinx_thumbnails dictionary can contain wildcards, which behave very similarly to the html_sidebars\textsuperscript{161} option.

The thumbnail files can be local image files somewhere in the source directory, but you’ll need to create at least one link (page 26) to them in order to copy them to the HTML output directory.

You can also use files from the _static directory (which contains all files in your html_static_path\textsuperscript{162}). If you want, you can also use files from the _images directory, which contains all notebook outputs.

To demonstrate this feature, we are creating an image file here:

```python
[1]: %matplotlib agg
```

```python
[2]: import matplotlib.pyplot as plt
```

```python
[3]: fig, ax = plt.subplots()  
  ax.plot([4, 8, 15, 16, 23, 42])  
  fig.savefig('a-local-file.png')
```

Please note that the previous cell doesn’t have any outputs, but it has generated a file named a-local-file.png in the notebook’s directory.

We have to create a link to this file (which is a good idea anyway): a-local-file.png.

Now we can use this file in our conf.py like this:

```python
nbsphinx_thumbnails = {  
  'gallery/thumbnail-from-conf-py': 'gallery/a-local-file.png',
}
```

Please note that the notebook name does not contain the .ipynb suffix.

\textsuperscript{161} https://www.sphinx-doc.org/en/master/usage/configuration.html#confval-html_sidebars  
\textsuperscript{162} https://www.sphinx-doc.org/en/master/usage/configuration.html#confval-html_static_path
12 Using toctree In A Notebook

In Sphinx-based documentation, there is typically a file called `index.rst` which contains one or more `toctree` directives. Those can be used to pull in further source files (which themselves can contain further `toctree` directives).

With `nbsphinx` it is possible to get a similar effect within a Jupyter notebook using the "nbsphinx-toctree" cell tag or cell metadata. Markdown cells with "nbsphinx-toctree" tag/metadata are not converted like "normal" Markdown cells. Instead, they are only scanned for links to other notebooks (or *.rst files and other Sphinx source files) and those links are added to a `toctree` directive. External links can also be used, but they will not be visible in the LaTeX output.

If there is a section title in the selected cell, it is used as `toctree` caption (but it also works without a title).

---

**Note**

All other content of such a cell is ignored!

---

If you are satisfied with the default settings, you can simply use "nbsphinx-toctree" as a cell tag.

Alternatively, you can store "nbsphinx-toctree" cell metadata. Use ...

```json
{
  "nbsphinx-toctree": {}
}
```

... for the default settings, ...

```json
{
  "nbsphinx-toctree": {
    "maxdepth": 2
  }
}
```

... for setting the `:maxdepth:` option, or ...

```json
{
  "nbsphinx-toctree": {
    "hidden": true
  }
}
```

... for setting the `:hidden:` option.

Of course, multiple options can be used at the same time, e.g.

```json
{
  "nbsphinx-toctree": {
    "maxdepth": 3,
    "numbered": true
  }
}
```

For more options, have a look a the Sphinx documentation\(^{163}\). All options can be used – except `:glob:`, which can only be used in `rst files` (page 54) and in `raw reST cells` (page 38).

\(^{163}\) https://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#directive-toctree

\(^{164}\) https://www.sphinx-doc.org/en/master/usage/restructuredtext/directives.html#directive-toctree
Note

In HTML output, a `toctree` cell generates an in-line table of contents (containing links) at its position in the notebook, whereas in the LaTeX output, a new (sub-)section with the actual content is inserted at its position. All content below the `toctree` cell will appear after the table of contents/inserted section, respectively. If you want to use the LaTeX output, it is recommended that you don’t add further cells below a `toctree` cell, otherwise their content may appear at unexpected places. Multiple `toctree` cells in a row should be fine, though.

The following cell is tagged with "nbsphinx-toctree" and contains a link to the notebook `yet-another.ipynb` (page 54) and an external link (which will only be visible in the HTML output). It also contains a section title which will be used as `toctree` caption (which also will only be visible in the HTML output).

12.1 Yet Another Notebook

This notebook is only here to show how (sub-)toctrees can be created with Markdown cell metadata. See `there` (page 53).

13 Normal reStructuredText Files

This is a normal RST file.

Note: Those still work!

13.1 Links to Notebooks (and Other Sphinx Source Files)

Links to Sphinx source files can be created like normal Sphinx hyperlinks, just using a relative path to the local file: `link` (page 47).

```rst
using a relative path to the local file: link_.
```

```rst
.. _link: subdir/a-notebook-in-a-subdir.ipynb
```

If the link text has a space (or some other strange character) in it, you have to surround it with backticks: a notebook link (page 47).

```rst
surround it with backticks: `a notebook link`_.
```

```rst
.. _a notebook link: subdir/a-notebook-in-a-subdir.ipynb
```

You can also use an anonymous hyperlink target, like this: `link` (page 47). If you have multiple of those, their order matters!

---


<sup>166</sup> [http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#anonymous-hyperlinks](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#anonymous-hyperlinks)
Finally, you can use Embedded URIs\textsuperscript{167}, like this \url{link} (page 47).

Note: These links should also work on Github and in other rendered reStructuredText pages.

Links to subsections are also possible by adding a hash sign (#) and the section title to any of the above-mentioned link variants. You have to replace spaces in the section titles by hyphens. For example, see this subsection (page 48).

For example, see this subsection_.

.. _subsection: subdir/a-notebook-in-a-subdir.ipynb#A-Sub-Section

### 13.2 Links to Notebooks, Ye Olde Way

In addition to the way shown above, you can also create links to notebooks (and other Sphinx source files) with \texttt{:ref:}\textsuperscript{168}. This has some disadvantages:

- It is arguably a bit more clunky.
- Because \texttt{:ref:} is a Sphinx feature, the links don’t work on Github and other rendered reStructuredText pages that use plain old docutils.

It also has one important advantage:

- The link text can automatically be taken from the actual section title.

A link with automatic title looks like this: \textit{Notebooks in Sub-Directories} (page 47).

\texttt{:ref:`/subdir/a-notebook-in-a-subdir.ipynb`}

But you can also provide your own link title (page 47).

\texttt{:ref:`your own link title </subdir/a-notebook-in-a-subdir.ipynb>`}

However, if you want to use your own title, you are probably better off using the method described above in \textit{Links to Notebooks (and Other Sphinx Source Files)} (page 54).

Links to subsections are also possible, e.g. \textit{A Sub-Section} (page 48) (the subsection title is used as link text) and \textit{alternative text} (page 48).

These links were created with:

\texttt{:ref:`/subdir/a-notebook-in-a-subdir.ipynb#A-Sub-Section`}
\texttt{:ref:`alternative text </subdir/a-notebook-in-a-subdir.ipynb#A-Sub-Section>`}

Note:

- The paths have to be relative to the top source directory and they have to start with a slash (/).
- Spaces in the section title have to be replaced by hyphens!

\textsuperscript{167} http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html\#embedded-uris-and-aliases
\textsuperscript{168} https://www.sphinx-doc.org/en/master/usage/restructuredtext/roles.html\#role-ref
13.3 Sphinx Directives for Info/Warning Boxes

Warning
This is an experimental feature! Its usage may change in the future or it might disappear completely, so don’t use it for now.

With a bit of luck, it will be possible (some time in the future) to create info/warning boxes in Markdown cells, see https://github.com/jupyter/notebook/issues/1292. If this ever happens, nbapphinx will provide directives for creating such boxes. For now, there are two directives available: nbinfo and nbwarning. This is how an info box looks like:

Note
This is an info box.
It may include nested formatting, even another info/warning box:

Warning: You should probably not use nested boxes!

13.4 Domain Objects

domain_object \( <foo> \)
This is just for testing domain object links. See this section (page 26).

Parameters  \( <foo> \) (str) – Example string parameter

13.5 Citations

You could use standard Sphinx citations\(^\text{169}\), but it might be more practical to use the \texttt{sphinxcontrib.bibtex}\(^\text{170}\) extension.

If you install and enable this extension, you can create citations like [PGH11]:

\texttt{:cite:`perez2011python`}

You can create similar citations in Jupyter notebooks with a special HTML syntax, see the section about citations in Markdown cells (page 21).

For those citations to work, you also need to specify a BibTeX file, as explained in the next section.

\(^{169}\) https://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#citations
\(^{170}\) https://sphinxcontrib-bibtex.readthedocs.io/
13.6 References

After installing and enabling (page 9) the sphinxcontrib.bibtex\(^{171}\) extension, you can create a list of references from a BibTeX file like this:

```bibtex
.. bibliography:: references.bib
```

Have a look at the documentation for all the available options.

The list of references may look something like this (in HTML output):

However, in the LaTeX/PDF output the list of references will not appear here, but at the end of the document. For a possible work-around, see https://github.com/mcmtroffaes/sphinxcontrib-bibtex/issues/156.

There is an alternative Sphinx extension for creating bibliographies: https://bitbucket.org/wnielson/sphinx-natbib/. However, this project seems to be abandoned (last commit in 2011).

13.7 Thumbnail Galleries

With nb\texttt{sphinx} you can create thumbnail galleries in notebook files as described in Creating Thumbnail Galleries (page 48).

If you like, you can also create such galleries in reST files using the nb\texttt{gallery} directive.

It takes the same parameters as the toctree\(^{174}\) directive.

**Note:** The notes regarding LaTeX in Creating Thumbnail Galleries (page 48) and Using toctree In A Notebook (page 53) also apply here!

The following example gallery was created using:

```py
.. nb\texttt{gallery}:
   :caption: This is a thumbnail gallery:
   :name: rst-gallery
   :glob: gallery/**-rst
```

The following section was generated from doc/gallery/uno-rst.ipynb .................................................. doc/gallery/uno-rst.ipynb ends here.

13.7.1 Dummy Notebook 1 for Gallery

This is a dummy file just to fill the gallery in the reST file (page 57).

The thumbnail image is assigned in conf.py.

The following section was generated from doc/gallery/uno-rst.ipynb .................................................. doc/gallery/uno-rst.ipynb ends here.

13.7.2 Dummy Notebook 2 for Gallery

This is a dummy file just to fill the gallery in the reST file (page 57).

The thumbnail image is assigned in conf.py.

\(^{171}\) https://sphinxcontrib-bibtex.readthedocs.io/

The source file is, for no particular reason, a Python script adhering to the `py:percent` format. It is parsed with the help of Jupyter\textsuperscript{175}, see Custom Notebook Formats (page 46).

[1]: from pathlib import Path

[2]: filename = 'due-rst.pct.py'

```python
print(Path(filename).read_text())
```

# %% [markdown]
# # Dummy Notebook 2 for Gallery
#
# This is a dummy file just to fill
# [the gallery in the reST file](../a-normal-rst-file.rst#thumbnail-galleries).
#
# The thumbnail image is assigned in [conf.py](../conf.py).

# %% [markdown]
# The source file is, for no particular reason, a Python script adhering to the `py:percent` format.
# It is parsed with the help of [Jupyter](https://jupyter.readthedocs.io/), see [Custom Notebook Formats](../custom-formats.ipynb).

# %%
from pathlib import Path

# %%
filename = 'due-rst.pct.py'

print(Path(filename).read_text())
```

---

The following section was generated from `doc/links.ipynb`

14 External Links

nbconvert

The official conversion tool of the Jupyter project. It can be used to convert notebooks to HTML, LaTeX and many other formats.

Its `--execute` flag can be used to automatically execute notebooks before conversion.

https://nbconvert.readthedocs.io/
https://github.com/jupyter/nbconvert

RunNotebook (notebook_sphinxext.py)

Notebooks can be included in `*.rst` files with a custom notebook directive. Uses nbconvert to execute notebooks and to convert the result to HTML.

No LaTeX support.

https://github.com/ngoldbaum/RunNotebook

There are some forks:

- https://bitbucket.org/yt_analysis/yt-doc/src/default/extensions/notebook_sphinxext.py

\textsuperscript{175} https://jupyter.readthedocs.io/

**nbsite**

Build a tested, sphinx-based website from notebooks.

https://nbsite.pyviz.org/

**ipypublish**

A workflow for creating and editing publication ready scientific reports and presentations, from one or more Jupyter Notebooks, without leaving the browser!

https://ipypublish.readthedocs.io/

https://github.com/chrisjsewell/ipypublish

**jupyter-book**

Create an online book with Jupyter Notebooks and Jekyll

https://jupyter.org/jupyter-book

https://github.com/jupyter/jupyter-book

**MyST-NB**

A collection of tools for working with Jupyter Notebooks in Sphinx. The primary tool this package provides is a Sphinx parser for ipynb files. This allows you to directly convert Jupyter Notebooks into Sphinx documents. It relies heavily on the `MyST parser <https://github.com/ExecutableBookProject/myst_parser>`_.

https://myst-nb.readthedocs.io/

https://github.com/ExecutableBookProject/MyST-NB

**nbinteract**

Create interactive webpages from Jupyter Notebooks

https://www.nbinteract.com/

https://github.com/SamLau95/nbinteract

**nb_pdf_template**

An extended nbconvert template for LaTeX output.

https://github.com/t-makaro/nb_pdf_template

**nb2plots**

Notebook to reStructuredText converter which uses a modified version of the matplotlib plot directive.

https://github.com/matthew-brett/nb2plots

**brole**

A Sphinx role for IPython notebooks

https://github.com/matthew-brett/brole

**Sphinx-Gallery**

https://sphinx-gallery.readthedocs.io/

**sphinx-nbexamples**

https://sphinx-nbexamples.readthedocs.io/
https://github.com/Chilipp/sphinx-nbexamples

nbsphinx-link
https://github.com/vidartf/nbsphinx-link

Uses nbsphinx, but supports notebooks outside the Sphinx source directory. See https://github.com/spatialaudio/nbsphinx/pull/33 for some limitations.

bookbook
Uses nbconvert to create a sequence of HTML or a concatenated LaTeX file from a sequence of notebooks.
https://github.com/takluyver/bookbook

jupyter-sphinx
Jupyter Sphinx is a Sphinx extension that executes embedded code in a Jupyter kernel, and embeds outputs of that code in the output document. It has support for rich output such as images, LaTeX math and even javascript widgets.
https://jupyter-sphinx.readthedocs.io/
https://github.com/jupyter/jupyter-sphinx

DocOnce

Converting Notebooks to reStructuredText
https://github.com/perrette/dimarray/blob/master/docs/scripts/nbconvert_to_rst.py
https://gist.github.com/hadim/16e29b5848672e2e497c (not available anymore)
https://sphinx-ipynb.readthedocs.io/

Converting reStructuredText to Notebooks
https://github.com/nthiery/rst-to-ipynb
https://github.com/QuantEcon/sphinxcontrib-jupyter

Converting Notebooks to HTML for Blog Posts
http://dongweiming.github.io/divingintoipynb_nikola/posts/nbconvert.html
https://github.com/getpelican/pelican-plugins/blob/master/liquid_tags/notebook.py

Further Posts and Issues
https://github.com/ipython/ipython/issues/4936
https://mail.scipy.org/pipermail/ipython-user/2013-December/013490.html (not available anymore)

------------------------------------------------------------------------------------- doc/links.ipynb ends here.
15 Contributing

If you find bugs, errors, omissions or other things that need improvement, please create an issue or a pull request at http://github.com/spatialaudio/nbsphinx/. Contributions are always welcome!

15.1 Development Installation

Instead of pip-installing the latest release from PyPI\textsuperscript{176}, you should get the newest development version (a.k.a. “master”) from Github\textsuperscript{177}:

\begin{verbatim}
git clone https://github.com/spatialaudio/nbsphinx.git
cd nbsphinx
python3 -m pip install -e . --user
\end{verbatim}

This way, your installation always stays up-to-date, even if you pull new changes from the Github repository. If you have only Python 3 installed, you might have to use the command `python` instead of `python3`. When installing `nbsphinx` this way, you can also quickly check other Git branches (in this example the branch is called “another-branch”):

\begin{verbatim}
git checkout another-branch
\end{verbatim}

When you run Sphinx now, it automatically uses the version “another-branch” of `nbsphinx`. If you want to go back to the “master” branch, use:

\begin{verbatim}
git checkout master
\end{verbatim}

To get the latest changes from Github, use:

\begin{verbatim}
git pull --ff-only
\end{verbatim}

15.2 Building the Documentation

If you make changes to the documentation, you should create the HTML pages locally using Sphinx and check if they look OK.

Initially, you might need to install a few packages that are needed to build the documentation:

\begin{verbatim}
python3 -m pip install -r doc/requirements.txt --user
\end{verbatim}

To (re-)build the HTML files, use:

\begin{verbatim}
python3 setup.py build_sphinx
\end{verbatim}

If you want to check the LaTeX output, use:

\begin{verbatim}
python3 setup.py build_sphinx -b latex
\end{verbatim}

Again, you’ll probably have to use `python` instead of `python3`. The generated files will be available in the directories `build/sphinx/html/` and `build/sphinx/latex/`, respectively.

\textsuperscript{176} https://pypi.org/project/nbsphinx/
\textsuperscript{177} https://github.com/spatialaudio/nbsphinx/
15.3 Testing

Unfortunately, the currently available automated tests are very limited. Contributions to improve the
testing situation are of course also welcome!

The nbsphinx documentation also serves as a test case. However, the resulting HTML/LaTeX/PDF
files have to be inspected manually to check whether they look as expected.

Sphinx’s warnings can help spot problems, therefore it is recommended to use the `-W` flag to turn
Sphinx warnings into errors while testing:

```
python3 setup.py build_sphinx -W
```

This flag is also used for continuous integration on Travis-CI (see the file `.travis.yml`) and CircleCI
(see the file `.circleci/config.yml`).

Sphinx has a linkcheck builder that can check whether all URLs used in the documentation are still
valid. This is also part of the continuous integration setup on CircleCI.

16 Version History

Version 0.5.1 (2020-01-28):

- This will be the last release supporting Python 2.x!
- Support for https://github.com/choldgraf/sphinx-copybutton
- Executed notebooks are now saved in the HTML output directory

Version 0.5.0 (2019-11-20):

- Automatic support for Jupyter widgets, customizable with nbsphinx_widgets_path (and
  nbsphinx_widgets_options)

Version 0.4.3 (2019-09-30):

- Add option nbsphinx_requirejs_path (and nbsphinx_requirejs_options)

Version 0.4.2 (2019-01-15):

- Re-establish Python 2 compatibility (but the clock is ticking . . .)

Version 0.4.1 (2018-12-16):

- Fix issue #266

Version 0.4.0 (2018-12-14):

- Support for “data-cite” HTML tags in Markdown cells
- Add option nbsphinx_custom_formats
- LaTeX macros \nbsphinxstartnotebook and \nbsphinxstopnotebook
- Support for cell attachments
- Add options nbsphinx_input_prompt and nbsphinx_output_prompt
- Re-design LaTeX output of code cells, fix image sizes

Version 0.3.5 (2018-09-10):

- Disable nbconvert version 5.4 to avoid issue #878

Version 0.3.4 (2018-07-28):

178 https://github.com/jupyter/nbconvert/issues/878
- Fix issue #196 and other minor changes

**Version 0.3.3 (2018-04-25):**
- Locally linked files are only copied for Jupyter notebooks (and not anymore for other Sphinx source files)

**Version 0.3.2 (2018-03-28):**
- Links to local files are rewritten for all Sphinx source files (not only Jupyter notebooks)

**Version 0.3.1 (2018-01-17):**
- Enable notebook translations (NB: The use of reST strings is temporary!)

**Version 0.3.0 (2018-01-02):**
- Add options `nbsphinx_prolog` and `nbsphinx_epilog`
- Links from `.rst` files to notebooks have to start with a slash

**Version 0.2.18 (2017-12-03):**
- Fix issue #148

**Version 0.2.17 (2017-11-12):**
- Fix issue #146

**Version 0.2.16 (2017-11-07):**
- Fix issue #142

**Version 0.2.15 (2017-11-03):**
- Links to subsections are now possible in all source files

**Version 0.2.14 (2017-06-09):**
- Add option `nbsphinx_kernel_name`

**Version 0.2.13 (2017-01-25):**
- Minor fixes

**Version 0.2.12 (2016-12-19):**
- Basic support for widgets
- CSS is now “responsive”, some new CSS classes

**Version 0.2.11 (2016-11-19):**
- Minor fixes

**Version 0.2.10 (2016-10-16):**
- Enable JavaScript output cells

**Version 0.2.9 (2016-07-26):**
- Add option `nbsphinx_prompt_width`

**Version 0.2.8 (2016-05-20):**
- Add options `nbsphinx_execute` and `nbsphinx_execute_arguments`
- Separate “display priority” for HTML and LaTeX

**Version 0.2.7 (2016-05-04):**
- Special CSS tuning for `sphinx_rtd_theme`
- Replace info/warning `<div>` elements with `nbinf/nbwarning`
Version 0.2.6 (2016-04-12):

• Support for LaTeX math environments in Markdown cells
• Add options nbsphinx_timeout and nbsphinx_codecell_lexer

Version 0.2.5 (2016-03-15):

• Add option nbsphinx_allow_errors to globally ignore exceptions
• Separate class nbsphinx.Exporter

Version 0.2.4 (2016-02-12):

• Support for “nbsphinx-toctree” cell metadata

Version 0.2.3 (2016-01-22):

• Links from notebooks to local files can now be used

Version 0.2.2 (2016-01-06):

• Support for links to sub-sections in other notebooks

Version 0.2.1 (2016-01-04):

• No need to mention source_suffix and source_parsers in conf.py

Version 0.2.0 (2015-12-27):

• Add support for allow_errors and hidden metadata
• Add custom reST template
• Add nbinput and nboutput directives with HTML+CSS and LaTeX formatting
• Turn nbsphinx into a Sphinx extension

Version 0.1.0 (2015-11-29): Initial release

References


72 https://doi.org/10.3233/978-1-61499-649-1-87
73 https://doi.org/10.1109/MCSE.2010.119