Effective Programming Practices for Economists

13. Documenting (the code of) research projects

Hans-Martin von Gaudecker

Department of Economics, Universität Bonn

universitätbonn

At the end of this lecture you are able to ...

- Appreciate the benefits of (code) documentation
- Create LaTeX and HTML documents using Sphinx and Waf
- Read and write reStructuredText documents
- Auto-document Python code using Sphinx
- Use Sphinx for documenting code in various languages

Licensed under the Creative Commons Attribution License

Documenting your code is indispensable

- Required by journals
- Required if anyone should use your code ...
  - Mandatory for putting user-written commands on SSC
  - Switching the type of markup is close to trivial
- ... and to liberate yourself from the most obvious questions
- You will be “just another user” in some time yourself

Licensed under the Creative Commons Attribution License

Why yet another document?

- Why not simply look at the code to understand it?

  Clean code $=$ perfect documentation (e.g. Eddins, 2008)?

- Far too fine-grained
  - Plow through implementations for understanding interfaces?
  - Lots of irrelevant information
  - Clean code and documentation are complements

- Designed to be understood by computers, not by humans

Licensed under the Creative Commons Attribution License

Why yet another document?

- Put it in the paper? Should contain all relevant information
- Most readers care about results; editors about journal space
  - Common mistake by young authors to describe too many details of **how** they were solving problems
  - Information you can provide is far too abstract to be usable as an entry point to the code
- Often you want to closely weave code and documentation
  - Standalone LaTeX not well suited
  - Other typesetting programs much less so

Licensed under the Creative Commons Attribution License

How to increase the chances that documentation is actually written?

- Problem of implementation:
  - Easy to agree that it makes perfect sense
  - Researchers hardly ever do it (properly)

⇒ Need a tool to make things easy

- Enter Sphinx – de-facto standard for Python code
- Matlab: Use Sphinx or m2html

  `http://www.artefact.tk/software/matlab/m2html/`
- R: Use Sphinx or Sweave

  `http://www.stat.uni-muenchen.de/~leisch/Sweave/`
- Stata: Use Sphinx + line-based inclusion of code

Licensed under the Creative Commons Attribution License

Taxonomy of documentation types

- Tutorials
  - Relevant when other people use your software (estimator?)
  - Far too often neglected
- API documentation (application programming interface)
  - Describes how program interacts with outside world
  - Keep interface definitions and their documentation close
  - Which brings us back to docstrings
- More on types of documentation: `http://blog.dexy.it/250`

Licensed under the Creative Commons Attribution License

What is Sphinx?

- Document preparation system
  - Think simpler LaTeX geared towards code documentation
  - Direct output to PDF (via LaTeX) and HTML
- Much simpler markup language (reStructuredText)
  - Similar ideas as Wiki languages, but more powerful
- Extract docstrings from Python modules
- Import code from other types of source code
  - Complete files, line-number or line-content based sections
- Automatically create indices and search functionality

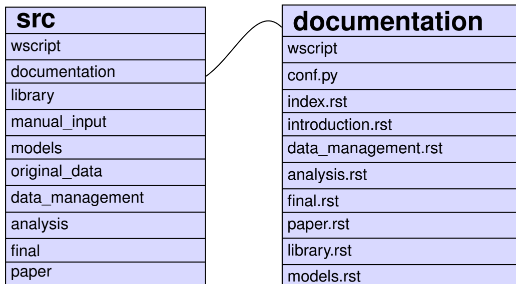
Licensed under the Creative Commons Attribution License

Getting started with Sphinx

- Focus on how to use Sphinx within project template
  - More general introduction is at `http://sphinx-doc.org/`
  - Excellent page
- Starting points for Sphinx are a configuration script ...
  - `conf.py`
- ... and a main reST file defining the basic structure
  - `index.rst`
- Waf task generators create PDF and HTML documents

Licensed under the Creative Commons Attribution License

Structure of the documentation directory



Licensed under the Creative Commons Attribution License

Important overall options in `conf.py`

```
# Add any Sphinx extension module names here, as strings.
# They can be extensions coming with Sphinx (named "sphinx.ext.*")
# or your custom ones.
extensions = [
    "sphinx.ext.autodoc",
    "sphinx.ext.viewcode",
    'sphinxcontrib.bibtex',
    "sphinx.ext.mathjax"
]

# Add any paths that contain templates here, relative to this directory.
templates_path = ["_templates"]

# The suffix of source filenames.
source_suffix = ".rst"

# The encoding of source files.
source_encoding = "utf-8"

# The master toctree document.
master_doc = "index"

# General information about the project.
project = u"TTT"
copyright = u"2013-, NNN"
```

Licensed under the Creative Commons Attribution License

Important HTML-specific options in conf.py

```
# -- Options for HTML output ---------------------------------------------------

# The theme to use for HTML and HTML Help pages.  See the documentation for
# a list of builtin themes.
html_theme = "haiku"

# This is the file name suffix for HTML files (e.g. ".xhtml").
html_file_suffix = ".html"

# Output file base name for HTML help builder.
htmlhelp_basename = "somedoc"
```

Licensed under the Creative Commons Attribution License

Important LaTeX-specific options in conf.py

```
# -- Options for LaTeX output ---------------------------------------------------

latex_elements = {
    # The paper size ("letterpaper" or "a4paper").
    "papersize": "a4paper",

    # The font size ("10pt", "11pt" or "12pt").
    "pointsize": "11pt",

    # Remove the "Release ..." subtitle from the LaTeX frontpage.
    "releasename": ""

    # Additional stuff for the LaTeX preamble.
    #"preamble": "",
}

# Grouping the document tree into LaTeX files. List of tuples
# (source start file, target name, title, author, documentclass [howto/manual]).
# Usually the list will only consist of one tuple for the master file.
latex_documents = [(
    "index",
    "project_documentation.tex",
    "Documentation of the TTT project",
    "NNN",
    "manual"
)]
```

Licensed under the Creative Commons Attribution License

src.documentation.wscript.build

```
# Build the documentation in pdf (via LaTeX) and html format.
ctx(
    features='sphinx',
    builders=['html', 'latexpdf'],
    source='conf.py'
)
# Install only after the build has finished.
ctx.add_post_fun(post_install)
```

Licensed under the Creative Commons Attribution License

reStructuredText

- Yet another markup language?!?!?
- Dual objective of docstrings:
  - Form the basis of external documentation
  - Document the code in-place
- Which is why you want to use a **simple** markup language
  - Get a nicely formatted external documentation
  - Be able to read and understand quickly in-place

Licensed under the Creative Commons Attribution License

reStructuredText

- LaTeX is far too complex for such a purpose
- In reST, markup constructs and meaning are to some extent related

```
This is a section heading
===========================

followed by normal text
```

- The next slide has the master document (`index.rst`)

Licensed under the Creative Commons Attribution License

```rst
.. This the TTT project's documentation master file.

.. You can adapt this file completely to your liking,
.. but it should at least contain the `toctree` directive.


Welcome to the TTT project's documentation!
================================================

.. toctree::
    :maxdepth: 2

    introduction
    original_data
    data_management
    analysis
    final
    paper
    model_code
    model_specs
    library
    references
```

Licensed under the Creative Commons Attribution License

reST directives

- The toctree is an example of a reStructuredText directive, used to give special meaning to a block of text
- Syntax:

```
.. name:: arguments
    :options: option_value

    content
```

- Example: Documenting a hypothetical Python function:

```
.. function:: foo(x)
   :module: some_module_name

   Return a line of text input from the user.
```

- Watch the indentation!

Licensed under the Creative Commons Attribution License

```rst
Paragraphs
----------

The paragraph is the most basic block in a reST document. Paragraphs are
simply chunks of text separated by one or more blank lines. As in Python,
indentation is significant in reST, so all lines of the same paragraph
must be left-aligned to the same level of indentation.

    This is an indented paragraph for the sake of demonstration. You can
    enhance text in paragraphs using :ref:`inlinemarkup`.


.. _inlinemarkup:

Inline markup
-------------

The standard reST inline markup is quite simple: use

* one asterisk *for emphasis* (italics),
* two asterisks **for strong emphasis**  (boldface), and
* backquotes ``for code samples``.
```

Licensed under the Creative Commons Attribution License

HTML output

Paragraphs

The paragraph is the most basic block in a reST document. Paragraphs are simply chunks of text separated by one or more blank lines. As in Python, indentation is significant in reST, so all lines of the same paragraph must be left-aligned to the same level of indentation.

> This is an indented paragraph for the sake of demonstration. You can enhance text in paragraphs using *Inline markup*.

Inline markup

The standard reST inline markup is quite simple: use

- one asterisk *for emphasis* (italics),
- two asterisks **for strong emphasis** (boldface), and
- backquotes `for code samples`.

Licensed under the Creative Commons Attribution License

$\LaTeX$ output

1.1 Paragraphs

The paragraph is the most basic block in a reST document. Paragraphs are simply chunks of text separated by one or more blank lines. As in Python, indentation is significant in reST, so all lines of the same paragraph must be left-aligned to the same level of indentation.

> This is an indented paragraph for the sake of demonstration. You can enhance text in paragraphs using *Inline markup*.

1.2 Inline markup

The standard reST inline markup is quite simple: use

- one asterisk *for emphasis* (italics),
- two asterisks **for strong emphasis** (boldface), and
- backquotes `for code samples`.

Licensed under the Creative Commons Attribution License

Notes on simple markup constructs

- Create an arbitrary reference location by starting a line with two colons, followed by whitespace, an underscore, the reference name and ending in a double colon

```
.. _label_name:
```

- Create a link with a reference to that location

```
:ref:`label_name`
```

- Provide an explicit title for the link if the label was not placed before a section heading or a figure:

```
:ref:`Link title <label_name>`
```

Licensed under the Creative Commons Attribution License

Notes on simple markup constructs

- Create a bulleted list by starting a line with an asterisk
- For bullet points spanning multiple lines, keep the same level of indentation at each line
- For numbered list, use either the numbers themselves or the pound character for automatic enumeration

Licensed under the Creative Commons Attribution License

Notes on simple markup constructs

```
#. Hello,

   World.

#. This is the second item
```

1. Hello,
   World.
2. This is the second item

Licensed under the Creative Commons Attribution License

Documenting Python code
Preliminaries

- You need the `autodoc` extension enabled in `conf.py`
- Sphinx imports the module to be documented
  - Depends on the location from where you invoke Sphinx
  - You can adjust `sys.path` in `conf.py`
- No need to worry in the context of the project template, the project root is automatically on `sys.path`
- In the docstrings, you can use all the reST you want

Licensed under the Creative Commons Attribution License

src/analysis/schelling.py

```python
"""Run a Schelling (1969, :cite:`Schelling69`) segregation
model and store a list with locations by type at each cycle.

The scripts expects that a model name is passed as an
argument. The model name must correspond to a file called
``[model_name].json`` in the "IN_MODEL_SPECS" directory.

"""


import sys
import json
import logging
import pickle
import numpy as np

from src.model_code.agent import Agent
from bld.project_paths import project_paths_join as ppj


def setup_agents(model):
    """Load the simulated initial locations and return a list
    that holds all agents.

    """

...
```

Licensed under the Creative Commons Attribution License

Documenting Python code
Content of `src/documentation/analysis.rst`

```
.. _analysis:

************************************
Main model estimations / simulations
************************************

Documentation of the code in *src.analysis*. This is the cor


Schelling example
=================

.. automodule:: src.analysis.schelling
    :members:
```

Licensed under the Creative Commons Attribution License

Documenting Python code
Resulting html output

Main model estimations / simulations

Documentation of the code in *src.analysis*. This is the core of the project.

Schelling example

Run a Schelling (1969, [1]) segregation model and store a list with locations by type at each cycle.

The scripts expects that a model name is passed as an argument. The model name must correspond to a file called `[model_name].json` in the "IN_MODEL_SPECS" directory.

`run_analysis`(*agents*, *model*) [source]

Given an initial set of *agents* and the *model*'s parameters, return a list of dictionaries with *type: N x 2* items.

`setup_agents`(*model*) [source]

Load the simulated initial locations and return a list that holds all agents.

Licensed under the Creative Commons Attribution License

Notes on autodoc markup constructs

- The `automodule` directive displays the docstring
- Add the `members` option to also extract the specified members' function signatures and docstrings
- More autodoc directives:

  http://sphinx-doc.org/ext/autodoc.html
- Autodoc generates labels that you can reference with

```
:mod:`package.module`
:func:`package.module.function`
:class:`package.module.class`
```

- Similar behaviour with out the autodoc feature:

```
.. py:function:: exponent(x, y)

    Description of exponent function
```

Licensed under the Creative Commons Attribution License

Including code snippets

- End a paragraph with `::` and the next paragraph(s), if indented, will be interpreted as (Python) code
- Change highlighting language for the rest of the document:

```
.. highlight:: c
```

- Change highlighting language for the next block:

```
.. code-block:: c

        Some C code
```

- Use any pygments lexer: `http://pygments.org/docs/lexers/`
- Matlab, R, ...
- More details: `http://sphinx-doc.org/markup/code.html`

Licensed under the Creative Commons Attribution License

Including “docstrings” from other languages

```
/*

Estimate a simple survival time model.

Important: The do-file requires two arguments to be passed

    1. the name of the file itself (for keeping the log)
    2. the name of a file in ${OUT_MODELS} containing globals
       definitions for a model (this will generally be derived
       from a json-file in ${IN_MODELS})

*/


// Header do-file with path definitions, those end up in global macros.
include src/library/stata/project_paths
log using `"${PATH_OUT_ANALYSIS}/log/`1'_`2'.log"', replace


...
```

Licensed under the Creative Commons Attribution License

Including “docstrings” from other languages

```
**Example for documenting Stata code**

.. include:: ../analysis/streg_example.do
    :start-after: /*
    :end-before: */
```

Example for documenting Stata code

Estimate a simple survival time model.

Important: The do-file requires two arguments to be passed

1. the name of the file itself (for keeping the log)
2. the name of a file in ${OUT_MODELS} containing globals definitions for a model (this will generally be derived from a json-file in ${IN_MODELS})

Licensed under the Creative Commons Attribution License

Wrapping up

- In Sphinx / reST, it is extremely easy to pick up the basics
- Simple to write, easy to spot most markup constructs
- Complicated things annoying without syntax highlighting
- No excuses for not writing documentation anymore!!!

Licensed under the Creative Commons Attribution License

At the end of this lecture you are able to . . .

- Appreciate the benefits of (code) documentation
- Create LaTeX and HTML documents using Sphinx and Waf
- Read and write reStructuredText documents
- Auto-document Python code using Sphinx
- Use Sphinx for documenting code in various languages

Licensed under the Creative Commons Attribution License

References I

Eddins, Steve (2008). “Take Control of Your Code: Essential Software Development Tools for Engineers and Scientists”. Available at http://blogs.mathworks.com/images/steve/2008/handout_yale_20080425.pdf.

Licensed under the Creative Commons Attribution License

Acknowledgements

- This course is designed after and borrows a lot from the `Software Carpentry` course designed by `Greg Wilson` for scientists and engineers.
- The Software Carpentry course material is made available under a `Creative Commons Attribution License`, as is this course's material.

License for the course material

[Links to the full `legal` text and the source text for `this` page.] You are free:

- **to Share** to copy, distribute and transmit the work
- **to Remix** to adapt the work

Under the following conditions:

- **Attribution** You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

With the understanding that:

- **Waiver** Any of the above conditions can be waived if you get permission from the copyright holder.
- **Public Domain** Where the work or any of its elements is in the public domain under applicable law, that status is in no way affected by the license.
- **Other Rights** In no way are any of the following rights affected by the license:
  - Your fair dealing or fair use rights, or other applicable copyright exceptions and limitations;
  - The author's moral rights;
  - Rights other persons may have either in the work itself or in how the work is used, such as publicity or privacy rights.

Notice For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to `this` web page.