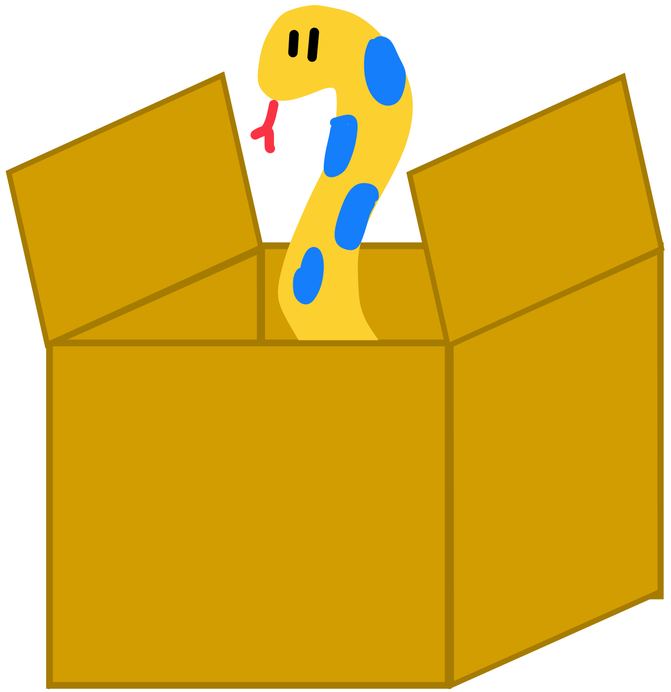


```
pip install mypackage
```

How to make a Python Package

Clara Burgard
Institut des Géosciences de l’Environnement

MC-Toolkit

09.02.2021

My personal motivation for tackling the subject…

During my PhD, I developed a tool to translate climate model output into brightness temperatures as could be measured by satellites

Observation operator

My personal motivation for tackling the subject…

During my PhD, I developed a tool to translate climate model output into brightness temperatures as could be measured by satellites

Sea-ice concentration
Sea-ice thickness
Surface temperature
…

Observation operator

A lot of interdependent functions

Brightness temperature

My personal motivation for tackling the subject…

During my PhD, I developed a tool to translate climate model output into brightness temperatures as could be measured by satellites

Sea-ice concentration
Sea-ice thickness
Surface temperature
…

Observation operator

A lot of interdependent functions

FUTURE ME AND POSSIBLE OTHER USERS:

Brightness temperature

My personal motivation for tackling the subject…

During my PhD, I developed a tool to translate climate model output into brightness temperatures as could be measured by satellites

Sea-ice concentration
Sea-ice thickness
Surface temperature
…

Observation operator

Brightness temperature

~~A lot of interdependent functions~~

FUTURE ME AND POSSIBLE OTHER USERS:

🤗

```
output = mypackage.compute(input)
```

+ possibility to install mypackage with pip and/or conda

How did I find out how to do it?

I wanted to make a python package but I did not know where to start. I got into a discussion with a former study colleague who is much deeper into Python and he suggested to help me!

How did I find out how to do it?

I wanted to make a python package but I did not know where to start. I got into a discussion with a former study colleague who is much deeper into Python and he suggested to help me!

Big acknowledgement to Philipp Sommer!

You can check his `psyplot` package for easy plotting of large climate model datasets (also on unstructured grids), see this DKRZ TechTalk for an explanation and application examples

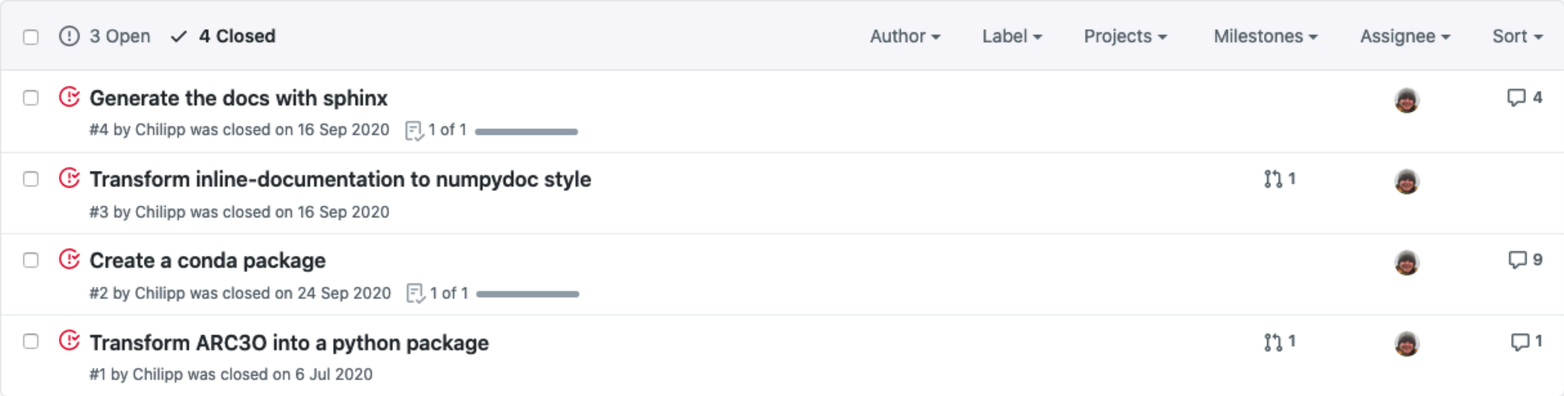
How did I find out how to do it?

I wanted to make a python package but I did not know where to start. I got into a discussion with a former study colleague who is much deeper into Python and he suggested to help me!

Big acknowledgement to Philipp Sommer!

You can check his `psyplot` package for easy plotting of large climate model datasets (also on unstructured grids), see this DKRZ TechTalk for an explanation and application examples

You can follow the procedure he took me through step-by-step on GitHub here



What to expect…

1. The basic recipe for a python package
2. The documentation
3. pip and conda install
4. Further refinements

What to expect…

1. The basic recipe for a python package
2. The documentation
3. pip and conda install
4. Further refinements

DISCLAIMER

I am not an expert and this is just one possible recipe!

First step - Having a plan - The 5 W’s and 1 H

Why?

What?

When?

How?

Where?

Who?

First step - Having a plan - The 5 W's and 1 H

Why?
What's the motivation for this package?

What?

When?

How?

Where?

Who?

First step - Having a plan - The 5 W's and 1 H

Why?
What's the motivation for this package?

When?
Are you just starting a new project or are you trying to make order in scripts that have accumulated?

Where?

What?

How?

Who?

First step - Having a plan - The 5 W's and 1 H

Why?
What's the motivation for this package?

What?

When?
Are you just starting a new project or are you trying to make order in scripts that have accumulated?

How?

Where?
Only for your personal use? On GitHub/Gitlab? Pip? Conda?

Who?

First step - Having a plan - The 5 W's and 1 H

Why?
What's the motivation for this package?

When?
Are you just starting a new project or are you trying to make order in scripts that have accumulated?

Where?
Only for your personal use? On GitHub/Gitlab? Pip? Conda?

What?
What should it do? What should be the input? What should be the output?

How?

Who?

First step - Having a plan - The 5 W's and 1 H

Why?
What's the motivation for this package?

When?
Are you just starting a new project or are you trying to make order in scripts that have accumulated?

Where?
Only for your personal use? On GitHub/Gitlab? Pip? Conda?

What?
What should it do? What should be the input? What should be the output?

How?
How should the package be structured? How should the functions be linked?

Who?

First step - Having a plan - The 5 W's and 1 H

Why?
What's the motivation for this package?

When?
Are you just starting a new project or are you trying to make order in scripts that have accumulated?

Where?
Only for your personal use? On GitHub/Gitlab? Pip? Conda?

What?
What should it do? What should be the input? What should be the output?

How?
How should the package be structured? How should the functions be linked?

Who?
What is your target "audience"? What would be a (good) name for the package?

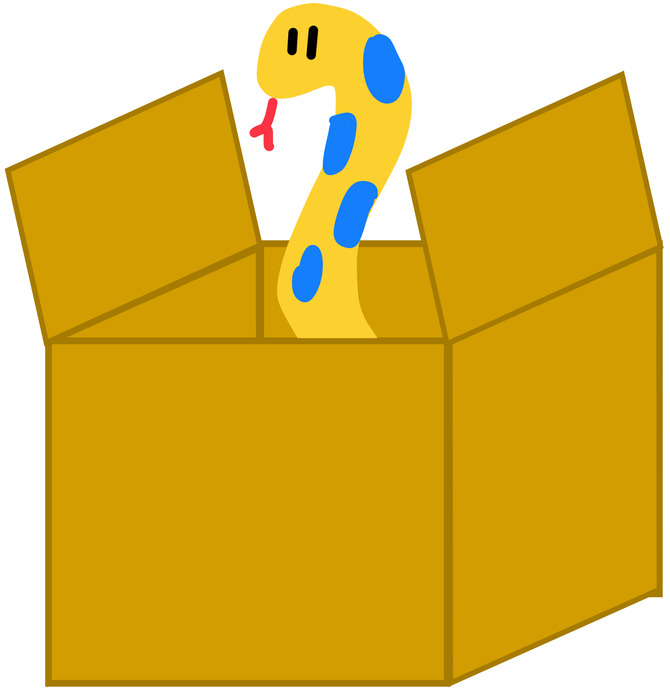
Let’s go!

1. The basic recipe for a python package
2. The documentation
3. pip and conda install
4. Further refinements

What is a package?

A package is a set of python functions that you want import at the beginning of a python script to be able to use them directly.

Famous examples: `numpy`, `xarray`, `matplotlib`...



The basic things needed for a python package

Nice tutorial, also step-by-step: https://packaging.python.org/tutorials/packaging-projects/

```
packaging_tutorial
├── LICENSE
├── README.md
├── example_pkg
│   └── __init__.py
├── setup.py
└── tests
```

The basic things needed for a python package

Nice tutorial, also step-by-step: https://packaging.python.org/tutorials/packaging-projects/

```
packaging_tutorial
├── LICENSE
├── README.md
├── example_pkg
│   └── __init__.py
├── setup.py
└── tests
```

Name of your repository

can but does not have to be the name of your package

The basic things needed for a python package

Nice tutorial, also step-by-step: https://packaging.python.org/tutorials/packaging-projects/

```
packaging_tutorial
├── LICENSE
├── README.md
├── example_pkg
│   └── __init__.py
├── setup.py
└── tests
```

License

tells users who install your package the terms under which they can use your package. For help picking a license, see https://choosealicense.com/.

The basic things needed for a python package

Nice tutorial, also step-by-step: https://packaging.python.org/tutorials/packaging-projects/

```
packaging_tutorial
├── LICENSE
├── README.md
├── example_pkg
│   └── __init__.py
├── setup.py
└── tests
```

Description of your package - can be seen when you open the repository on GitHub

md stands for markdown

The basic things needed for a python package

Nice tutorial, also step-by-step: https://packaging.python.org/tutorials/packaging-projects/

```
packaging_tutorial
├── LICENSE
├── README.md
├── example_pkg
│   └── __init__.py
├── setup.py
└── tests
```

Description of your package - can be seen when you open the repository on GitHub
md stands for markdown

ClimateClara Corrected missing word in index.rst `5db0751` on 18 Nov 2020 **82** commits

arc3o	added an option to define pool worker number	5 months ago
docs	Corrected missing word in index.rst	3 months ago
.gitignore	Merge branch 'master' into prepare_docs	5 months ago
LICENSE	Initial commit	10 months ago
MANIFEST.in	finished first draft of setup.py and created MANIFEST.in	7 months ago
README.rst	corrected missing word in README.rst	3 months ago
readthedocs.yml	worked on structure for the documentation files	6 months ago
requirements.txt	uncommented pathos in requirements.txt	5 months ago
setup.py	Merge branch 'master' into prepare_docs	5 months ago

README.rst

ARC3O

docs | docs passing

Welcome to the documentation about the Arctic Ocean Observation Operator!

The Arctic Ocean Observation Operator (ARC3O) computes brightness temperatures at 6.9 GHz, vertical polarization, based on climate model output. More information about the motivation, structure and evaluation can be found in Burgard et al., 2020a and Burgard et al., 2020b.

Currently, it is customized for output of the Max Planck Institute Earth System Model but can be used for other models if the variable names are changed accordingly in the ARC3O functions.

You can access the detailed documentation here: https://arc3o.readthedocs.io/

How to cite ARC3O

The detailed description and evaluation of ARC3O is found in Burgard et al., 2020b and should therefore, when used, be cited as follows:

Burgard, C., Notz, D., Pedersen, L. T., and Tonboe, R. T. (2020): "The Arctic Ocean Observation Operator for 6.9 GHz (ARC3O) – Part 2: Development and evaluation", *The Cryosphere*, 14, 2387–2407, https://doi.org/10.5194/tc-14-2387-2020.

The basic things needed for a python package

Nice tutorial, also step-by-step: https://packaging.python.org/tutorials/packaging-projects/

```
packaging_tutorial
├── LICENSE
├── README.md
├── example_pkg
│   └── __init__.py
├── setup.py
└── tests
```

Core of your package, folder with the functions
must be the name of your package

The basic things needed for a python package

Nice tutorial, also step-by-step: https://packaging.python.org/tutorials/packaging-projects/

```
packaging_tutorial
├── LICENSE
├── README.md
├── example_pkg
│   └── __init__.py
├── setup.py
└── tests
```

Required to import the directory as a package
can be empty or contain the top level functions

__init__.py - a short excourse

Required to import the directory as a package: can be empty or contain the top level functions

__init__.py	Merge branch 'master' into prepare_docs	5 months ago
core_functions.py	added an option to define pool worker number	5 months ago
mask_functions.py	Removed ana. in front of is_summer function - remnant of earlier vers...	5 months ago
memls_functions_2D.py	Apply suggestions from code review	5 months ago
profile_functions.py	Apply suggestions from code review	5 months ago

__init__.py - a short excourse

Required to import the directory as a package: can be empty or contain the top level functions

__init__.py	Merge branch 'master' into prepare_docs	5 months ago
core_functions.py	added an option to define pool worker number	5 months ago
mask_functions.py	Removed ana. in front of is_summer function - remnant of earlier vers...	5 months ago
memls_functions_2D.py	Apply suggestions from code review	
profile_functions.py	Apply suggestions from code review	

master arc3o / arc3o / __init__.py / <> Jump to

ClimateClara Merge branch 'master' into prepare_docs

1 contributor

13 lines (8 sloc) | 351 Bytes

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from .core_functions import satsim_complete_parallel
from .core_functions import satsim_complete_1month
from .core_functions import new_outputpath
from .core_functions import prep_time


__all__ = ['satsim_complete_parallel', 'satsim_complete_1month', 'new_outputpath', 'prep_time']

__version__='0.1'
```

Calling a function from your package normally:
`mypackage.<sub_file>.function()`

For functions that should be easily accessible, import them in the __init__.py file, then:
`mypackage.function()`

The basic things needed for a python package

Nice tutorial, also step-by-step: https://packaging.python.org/tutorials/packaging-projects/

```
packaging_tutorial
├── LICENSE
├── README.md
├── example_pkg
│   └── __init__.py
├── setup.py
└── tests
```

Required
contains all important information needed for python to set up the package. Basic explanation here.
Example here.

The basic things needed for a python package

Nice tutorial, also step-by-step: https://packaging.python.org/tutorials/packaging-projects/

```
packaging_tutorial
├── LICENSE
├── README.md
├── example_pkg
│   └── __init__.py
├── setup.py
└── tests
```

Required
contains all important information needed for python to set up the package. Basic explanation here.
Example here.

```python
import setuptools
import pathlib
import os

here = pathlib.Path(__file__).parent.resolve()

# Get the long description from the README file
long_description = open(os.path.join(here, 'README.rst'), encoding='utf-8').read()

setuptools.setup(

    #The project's name
    name='arc3o',

    #The project's version
    version='0.1',

    #The project's metadata
    author='Clara Burgard',
    author_email='clara.burgard@gmail.com',
    description='An observation operator for the Arctic Ocean for 6.9 GHz',
    long_description=long_description,

    #The project's main homepage.
    url='https://github.com/ClimateClara/arc3o',

    #The project's license
    license='GPL-3.0',

    packages=setuptools.find_packages(exclude=['docs', 'tests*', 'examples']),

    classifiers=[
        'Development Status :: 4 - Beta',
        'Intended Audience :: Science/Research',
        'License :: OSI Approved :: GNU General Public License v3 (GPLv3)',
        'Operating System :: OS Independent',
        'Programming Language :: Python :: 3 :: Only',
    ],

    project_urls={
        'Source': 'https://github.com/ClimateClara/arc3o',
        'Tracker': 'https://github.com/ClimateClara/arc3o/issues',
    #    'Documentation': 'https://arc3o.readthedocs.io',
      },

    keywords='earth-sciences climate-modeling sea-ice arctic oceanography remote-sensing',

    python_requires='>=3.5',

    install_requires=[
          'numpy',
          'xarray',
          'pandas',
          'tqdm',
          'pathos'
      ],
```

The basic things needed for a python package

Nice tutorial, also step-by-step: https://packaging.python.org/tutorials/packaging-projects/

```
packaging_tutorial
├── LICENSE
├── README.md
├── example_pkg
│   └── __init__.py
├── setup.py
└── tests
```

For tests

not 100% necessary, we'll come back to it in the end

pip install your own package! - developer mode

Does your repository contain all the files below?

```
packaging_tutorial
├── LICENSE
├── README.md
├── example_pkg
│   └── __init__.py
├── setup.py
└── tests
```

pip install your own package! - developer mode

Does your repository contain all the files below?

```
packaging_tutorial
├── LICENSE
├── README.md
├── example_pkg
│   └── __init__.py
├── setup.py
└── tests
```

Congratulations! You have a python package! 🥳

pip install your own package! - developer mode

Does your repository contain all the files below?

```
packaging_tutorial
├── LICENSE
├── README.md
├── example_pkg
│   └── __init__.py
├── setup.py
└── tests
```

Congratulations! You have a python package! 🥳

If you want to use it just for personal use and continue modifying it:

`pip install -e .`

from inside repository (-e = editable mode)

pip install your own package! - developer mode

Does your repository contain all the files below?

```
packaging_tutorial
├── LICENSE
├── README.md
├── example_pkg
│   └── __init__.py
├── setup.py
└── tests
```

Congratulations! You have a python package! 

If you want to use it just for personal use and continue modifying it:

```
pip install -e .
```

from inside repository (-e = editable mode)

And in your scripts you can now:

```
import mypackage as mp
```

or, if you have subfiles for functions:

```
import mypackage.subfile1 as mp1
```

Think about future you and other possible users!

Why documentation?

Why documentation?

Let’s face it:
Will you remember each decision and detail behind your code in a few years time?

Also:
Have you ever wanted to use code from someone else?

Why documentation?

Let’s face it:
Will you remember each decision and detail behind your code in a few years time?

Also:
Have you ever wanted to use code from someone else?

Documentation is essential for the reproducibility and usage of your package!

Prepare the documentation of your actual code

Describe your function directly in the code (docstring), following standards. For example: using numpy style

Prepare the documentation of your actual code

Describe your function directly in the code (docstring), following standards. For example: using [numpy style]

```
def ro2epsd(roi,Ti,freq):

    """Compute real part of dielectric permittivity for dry snow

    This function computes the dielectric permittivity for dry snow from density.

    Parameters
    ----------
    roi: np.array or xarray.DataArray
        snow density in g/cm3
    Ti: np.array or xarray.DataArray
        snow temperature in K
    freq: float
        frequency in GHz

    Returns
    -------
    epsi: np.array or xarray.DataArray
        real part of dielectric permittivity of dry snow
    epsii: np.array or xarray.DataArray
        imaginary part of dielectric permittivity of dry snow

    Notes
    -----
    This function is part of the original MEMLS developed by the Institute of Applied Physics,
    University of Bern, Switzerland. A description of that model version can be found in :cite:`wiesmann98`
    and :cite:`wiesmann99`. It was translated to Python and adapted for multi-dimensional input by `C. Burgard <http://www.github.com/ClimateClara>`_
    to be used in ARC3O.
    """
```

Documentation recipe (1)

You can use Sphinx. Sphinx is a *documentation generator* or a tool that translates a set of plain text source files into various output formats, automatically producing cross-references, indices, etc.

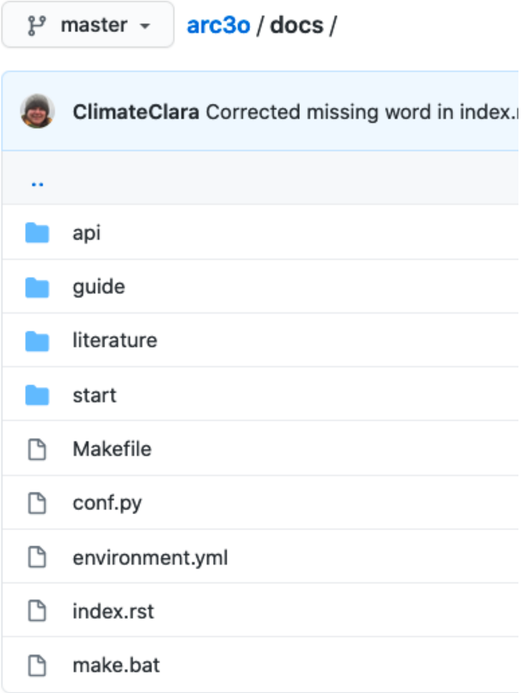
Documentation recipe (1)

You can use Sphinx. Sphinx is a *documentation generator* or a tool that translates a set of plain text source files into various output formats, automatically producing cross-references, indices, etc.

- Create a new folder called docs (./my_repository/docs)
- Get started with Sphinx-quickstart. It will generate the basic structure needed for the documentation.
- Include autodoc and intersphinx-mapping and some other parameters to conf.py (research what you need)
- Automatically create the API (Application Programming Interface, i.e. your functions) description based on the docstrings presented before using sphinx-autogen and include the API reference file in the `toctree` of your `index.rst` file.

Documentation recipe (2)

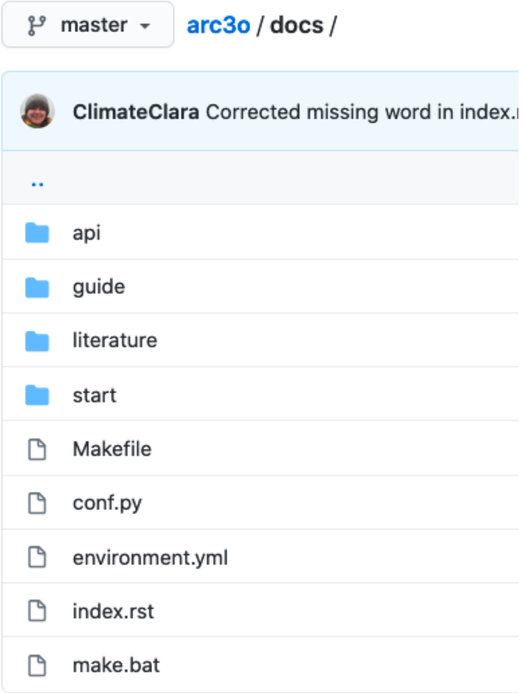
Add more general information about your package (e.g. an "About" or "How to install" section) and adapt your `index.rst` file.



Documentation recipe (2)

```
.. arc3o documentation master file, created by
   sphinx-quickstart on Mon Aug 10 11:47:09 2020.
   You can adapt this file completely to your liking, but it should at least
   contain the root `toctree` directive.

Welcome to the documentation about the Arctic Ocean Observation Operator!
=========================================================================

The Arctic Ocean Observation Operator (ARC3O) computes brightness temperatures at 6.9 GHz,
vertical polarization, based on climate model output. More information about the motivation,
structure and evaluation can be found in `Burgard et al., 2020a`_ and `Burgard et al., 2020b`_.

Currently, it is customized for output of the Max Planck Institute Earth System Model but can be
used for other models if the variable names are changed accordingly in the ARC3O functions.

Documentation
-------------
.. toctree::
   :maxdepth: 2
   :caption: Getting started:

   start/about
   start/installation

.. toctree::
   :maxdepth: 2
   :caption: User's Guide:

   guide/howtorun
   guide/workflow

.. toctree::
   :maxdepth: 2
   :caption: Help & References:

   api/arc3o
   literature/references
   literature/publications


How to cite ARC3O
-----------------

The detailed description and evaluation of ARC3O is found in `Burgard et al., 2020b`_ and should
therefore, when used, be cited as follows:

Burgard, C., Notz, D., Pedersen, L. T., and Tonboe, R. T. (2020): "The Arctic Ocean Observation Ope

.. _`Burgard et al., 2020a`: https://tc.copernicus.org/articles/14/2369/2020/
.. _`Burgard et al., 2020b`: https://tc.copernicus.org/articles/14/2387/2020/


Indices and tables
==================

* :ref:`genindex`
* :ref:`modindex`
* :ref:`search`
```

Add more general information about your package (e.g. an “About” or “How to install” section) and adapt your `index.rst` file.



Documentation recipe (2)

```
.. arc3o documentation master file, created by
   sphinx-quickstart on Mon Aug 10 11:47:09 2020.
   You can adapt this file completely to your liking, but it should at least
   contain the root `toctree` directive.

Welcome to the documentation about the Arctic Ocean Observation Operator!
==========================================================================

The Arctic Ocean Observation Operator (ARC3O) computes brightness temperatures at 6.9 GHz,
vertical polarization, based on climate model output. More information about the motivation,
structure and evaluation can be found in `Burgard et al., 2020a`_ and `Burgard et al., 2020b`_.

Currently, it is customized for output of the Max Planck Institute Earth System Model but can be
used for other models if the variable names are changed accordingly in the ARC3O functions.

Documentation
-------------
.. toctree::
   :maxdepth: 2
   :caption: Getting started:

   start/about
   start/installation

.. toctree::
   :maxdepth: 2
   :caption: User's Guide:

   guide/howtorun
   guide/workflow

.. toctree::
   :maxdepth: 2
   :caption: Help & References:

   api/arc3o
   literature/references
   literature/publications


How to cite ARC3O
-----------------

The detailed description and evaluation of ARC3O is found in `Burgard et al., 2020b`_ and should
therefore, when used, be cited as follows:

Burgard, C., Notz, D., Pedersen, L. T., and Tonboe, R. T. (2020): "The Arctic Ocean Observation Ope

.. _`Burgard et al., 2020a`: https://tc.copernicus.org/articles/14/2369/2020/
.. _`Burgard et al., 2020b`: https://tc.copernicus.org/articles/14/2387/2020/


Indices and tables
==================

* :ref:`genindex`
* :ref:`modindex`
* :ref:`search`
```

Add more general information about your package (e.g. an “About” or “How to install” section) and adapt your `index.rst` file.

master arc3o / docs /

ClimateClara Corrected missing word in index.

..

- api
- guide
- literature
- start
- Makefile
- conf.py
- environment.yml
- index.rst
- make.bat

master arc3o / docs / guide /

ClimateClara added a remark about changing t

..

- ._Figure1.pdf
- Figure1.png
- howtorun.rst
- step1.rst
- step2.rst
- step3.rst
- step4.rst
- step5.rst
- workflow.rst

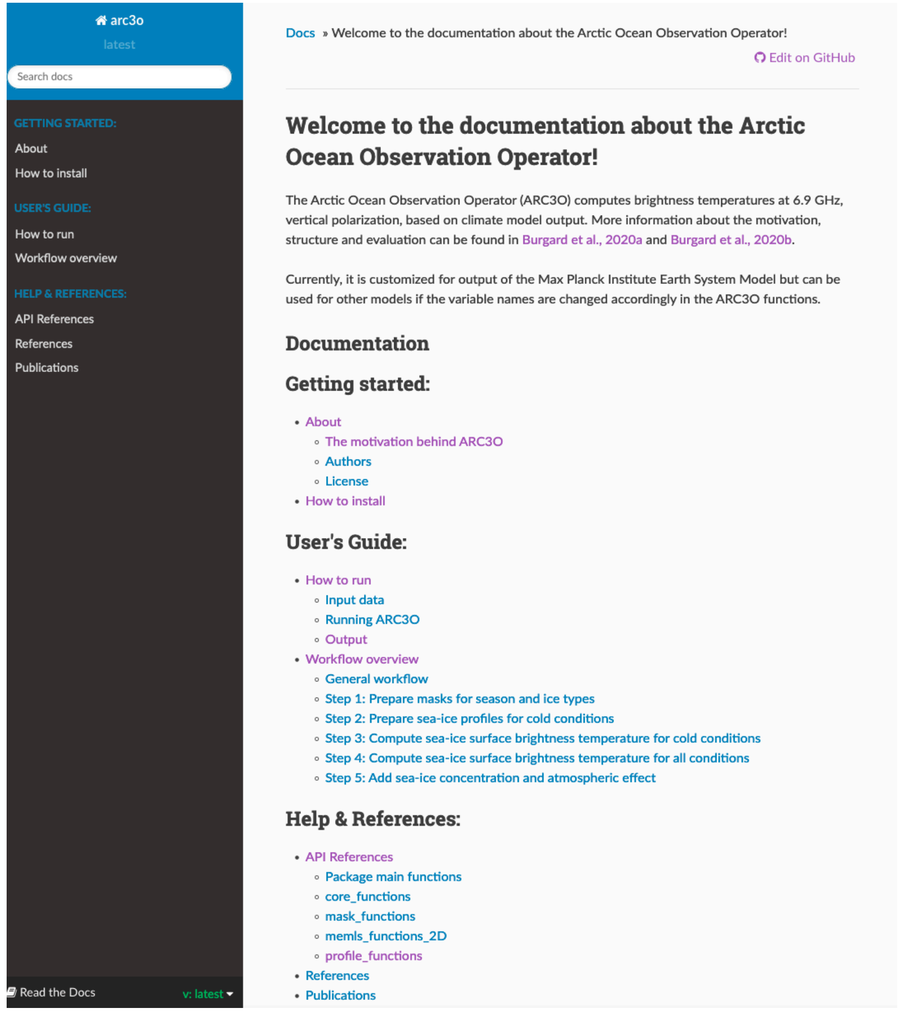
Documentation recipe (3)

Build the docs running `make html` in your `docs` folder

Documentation recipe (3)

Build the docs running `make html` in your `docs` folder

Open with a browser `docs/_build/html/index.html` to check it out

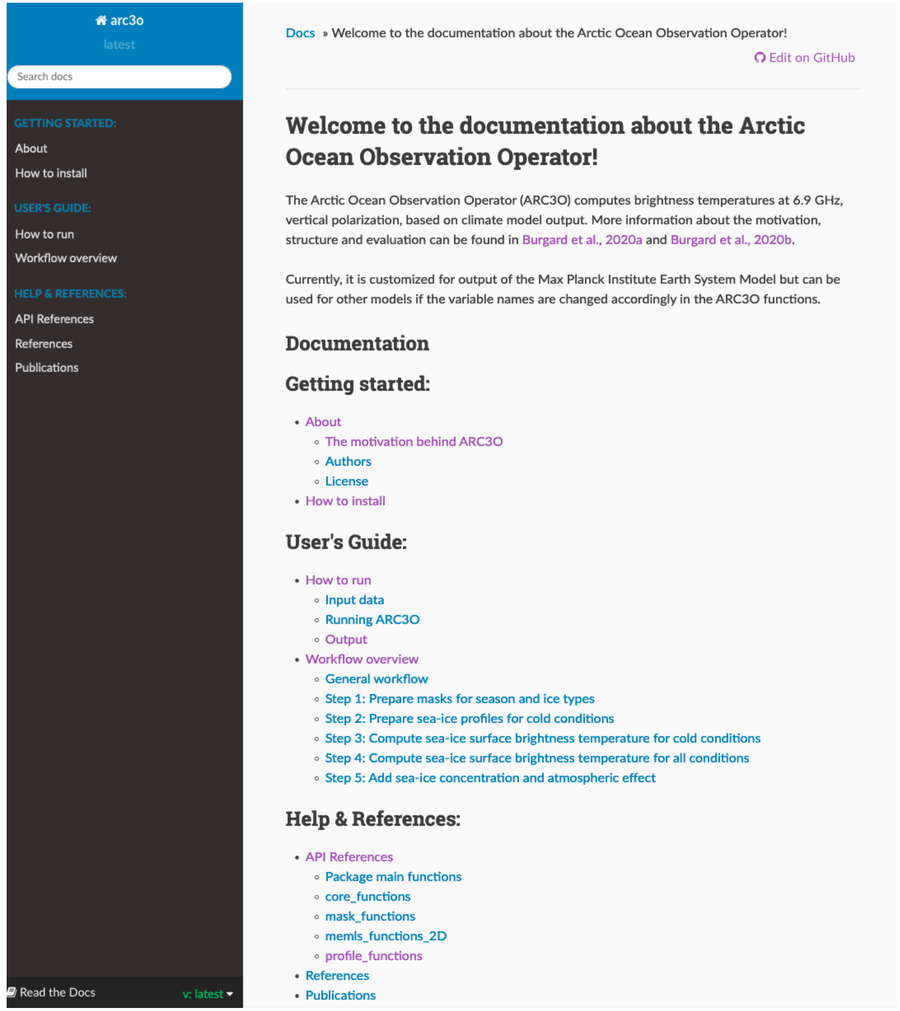


Documentation recipe (3)

Build the docs running `make html` in your `docs` folder

Open with a browser `docs/_build/html/index.html` to check it out

Once you are happy, you can upload it to readthedocs to make it publicly available, procedure is quite straightforward if you have your package repository in GitHub.



Getting real...

If you want pip and/or conda install, you need to go through pypi

When you're happy with your package and you have checked that everything works, you can upload it to pypi.org.

If you want pip and/or conda install, you need to go through pypi

When you're happy with your package and you have checked that everything works, you can upload it to pypi.org.

Here are the steps:

- Install `twine` (for example: `conda install twine` if you are working with conda)
- Run `python setup.py sdist` in your repository root
- Create an account on pypi.org.
- Upload the package by running `twine upload dist/mypackage*.tar.gz` in your repository root

Again, documentation can be found here.

If you want pip and/or conda install, you need to go through pypi

When you're happy with your package and you have checked that everything works, you can upload it to pypi.org.

Here are the steps:

- Install `twine` (for example: `conda install twine` if you are working with conda)
- Run `python setup.py sdist` in your repository root
- Create an account on pypi.org.
- Upload the package by running `twine upload dist/mypackage*.tar.gz` in your repository root

Again, documentation can be found here.

This is it! Now you can do: `pip install mypackage` 😃

Interested in making it a conda package as well?

The easiest is to go through conda-forge.

Interested in making it a conda package as well?

The easiest is to go through conda-forge.

Submit a Pull Request (PR) to the staged-recipes repository with your new conda build recipe. This is quite technical, please follow the steps here and here.

After your pull request has been accepted, you're done!

Interested in making it a conda package as well?

The easiest is to go through conda-forge.

Submit a Pull Request (PR) to the staged-recipes repository with your new conda build recipe.
This is quite technical, please follow the steps here and here.

After your pull request has been accepted, you're done!

This is it! Now you can do: `conda install -c conda-forge mypackage` 😀

Making it even more fancy

Possible ways of doing this even better

There are even more features, which I did not add. Philipp suggested the following:

Possible ways of doing this even better

There are even more features, which I did not add. Philipp suggested the following:

Tests

There are two different test types
- unit tests: they test one specific function (i.e. the smallest possible unit)
- integrational test: they test the entire workflow

Both are necessary (although the latter one are sometimes easier to think about and more important) but it's actually only a conceptual difference. I recommend to use pytest for both of them.

Possible ways of doing this even better

There are even more features, which I did not add. Philipp suggested the following:

Tests

There are two different test types
- unit tests: they test one specific function (i.e. the smallest possible unit)
- integrational test: they test the entire workflow

Both are necessary (although the latter one are sometimes easier to think about and more important) but it's actually only a conceptual difference. I recommend to use pytest for both of them.

Command-line tool

This requires some modification of setup.py and we should use argparse to generate an intuitive command-line interface. This can then even be rendered into the sphinx documentation using sphinx-argparse.

What you should take home…

Take home messages

Depending on your goals: a python package can go from simple to complicated

Great tool to start a project directly in a package structure: check out cookiecutter-science!

Take home messages

Depending on your goals: a python package can go from simple to complicated

Great tool to start a project directly in a package structure: check out cookiecutter-science!

Invest time in your documentation, it'll make other users and future you more likely to use the package

Take home messages

Depending on your goals: a python package can go from simple to complicated

Great tool to start a project directly in a package structure: check out cookiecutter-science!

Invest time in your documentation, it'll make other users and future you more likely to use the package

If I could do it, you can do it as well! 😎

Take home messages

Depending on your goals: a python package can go from simple to complicated

Great tool to start a project directly in a package structure: check out cookiecutter-science!

Invest time in your documentation, it'll make other users and future you more likely to use the package

If I could do it, you can do it as well! 😎

THANK YOU FOR YOUR ATTENTION!