

---

# **tobac**

***Release v2.0-dev***

**May 13, 2022**



---

## Contents

---

|          |                                   |           |
|----------|-----------------------------------|-----------|
| <b>1</b> | <b>Installation</b>               | <b>3</b>  |
| <b>2</b> | <b>*Data input and output</b>     | <b>5</b>  |
| <b>3</b> | <b>tobac themes</b>               | <b>7</b>  |
| <b>4</b> | <b>Analysis</b>                   | <b>9</b>  |
| <b>5</b> | <b>Plotting</b>                   | <b>11</b> |
| <b>6</b> | <b>Example notebooks</b>          | <b>13</b> |
| <b>7</b> | <b>API reference</b>              | <b>15</b> |
| 7.1      | Core Modules . . . . .            | 15        |
| 7.2      | Theme Modules: Tobac v1 . . . . . | 15        |



**tobac** is a Python package to identify, track and analyse clouds in different types of gridded datasets, such as 3D model output from cloud resolving model simulations or 2D data from satellite retrievals.

The software is set up in a modular way to include different algorithms for feature identification, tracking and analyses. In the current implementation, individual features are identified as either maxima or minima in a two dimensional time varying field. The volume/area associated with the identified object can be determined based on a time-varying 2D or 3D field and a threshold value. In the tracking step, the identified objects are linked into consistent trajectories representing the cloud over its lifecycle. Analysis and visualisation methods provide a convenient way to use and display the tracking results.

Version 1.2 of **tobac** and some example applications are described in a paper in the journal *Geoscientific Model Development* as:

Heikenfeld, M., Marinescu, P. J., Christensen, M., Watson-Parris, D., Senf, F., van den Heever, S. C., and Stier, P.: **tobac v1.2: towards a flexible framework for tracking and analysis of clouds in diverse datasets**, *Geosci. Model Dev.*, <https://doi.org/10.5194/gmd-12-4551-2019>, 2019.

The project is currently extended by several contributors to include additional workflows and algorithms using the same structure, syntax and data formats.



# CHAPTER 1

---

## Installation

---

tobac is now capable of working with both Python 2 and Python 3 (tested for Python 2.7, 3.6, 3.7 and 3.8) installations.

The easiest way is to install the most recent version of tobac via conda and the conda-forge channel:

```
` conda install -c conda-forge tobac `
```

This will take care of all necessary dependencies and should do the job for most users and also allows for an easy update of the installation by

```
` conda update -c conda-forge tobac `
```

You can also install conda via pip, which is mainly interesting for development purposed or to use specific development branches for the Github repository.

The following python packages are required (including dependencies of these packages):

*trackpy, scipy, numpy, iris, scikit-learn, scikit-image, cartopy, pandas, pytables*

**If you are using anaconda, the following command should make sure all dependencies are met and up to date:**

```
conda install -c conda-forge -y trackpy scipy numpy iris scikit-learn  
scikit-image cartopy pandas pytables
```

You can directly install the package directly from github with pip and either of the two following commands:

```
pip install --upgrade git+ssh://git@github.com/climate-processes/  
tobac.git  
  
pip install --upgrade git+https://github.com/climate-processes/tobac.  
git
```

You can also clone the package with any of the two following commands:

```
git clone git@github.com:climate-processes/tobac.git  
git clone https://github.com/climate-processes/tobac.git
```

and install the package from the locally cloned version (The trailing slash is actually necessary):

```
pip install --upgrade tobac/
```





---

### \*Data input and output

---

Input data for `tobac` should consist of one or more fields on a common, regular grid with a time dimension and two or more spatial dimensions. The input data should also include latitude and longitude coordinates, either as 1-d or 2-d variables depending on the grid used.

Interoperability with `Iris` and `pandas` is provided by the convenient functions allowing for a transformation between the data types. `xarray` `DataArrays` can be easily converted into `iris` cubes using `xarray`'s `to_iris()` method, while the `Iris` cubes produced as output of `tobac` can be turned into `xarray` `DataArrays` using the `from_iris()` method.

For the future development of the next major version of `tobac`, we are envisaging moving the basic data structures from `Iris` cubes to `xarray` `DataArrays` for improved computing performance and interoperability with other open-source software packages.

The output of the different analysis steps in `tobac` are output as either `pandas` `DataFrames` in the case of one-dimensional data, such as lists of identified features or cloud trajectories or as `Iris` cubes in the case of 2D/3D/4D fields such as cloud masks. Note that the dataframe output from tracking is a superset of the features dataframe.

(quick note on terms; “feature” is a detected object at a single time step. “cell” is a series of features linked together over multiple timesteps)

#### Overview of the output dataframe from `feature_detection`

- `Frame`: the index along the time dimension in which the feature was detected
- `hdim_1`, `hdim_2`, ...: the central index location of the feature in the spatial dimensions of the input data
- `num`: the number of connected pixels that meet the threshold for detection for this feature
- `threshold_value`: the threshold value that was used to detect this feature. When using `feature_detection_multithreshold` this is the max/min (depending on whether the threshold values are increasing (e.g. precip) or decreasing (e.g. temperature) with intensity) threshold value used.
- `feature`: a unique integer >0 value corresponding to each feature
- `time`: the date and time of the feature, in datetime format
- `timestr`: the date and time of the feature in string format
- `latitude`, `longitude`: the central lat/lon of the feature

- x,y, etc: these are the central location of the feature in the original dataset coordinates

**Also in the tracked output:**

- Cell: The cell which each feature belongs to. Is nan if the feature could not be linked into a valid trajectory
- time\_cell: The time of the feature along the tracked cell, in numpy.timedelta64[ns] format

The output from segmentation is an n-dimensional array produced by segmentation in the same coordinates of the input data. It has a single field, which provides a mask for the pixels in the data which are linked to each detected feature by the segmentation routine. Each non-zero value in the array provides the integer value of the feature which that region is attributed to.

Note that in future versions of tobac, it is planned to combine both output data types into a single hierarchical data structure containing both spatial and object information. Additional information about the planned changes can be found in the v2.0-dev project, as well as the tobac roadmap

## CHAPTER 3

---

### tobac themes

---

Starting from version 2.0 tobac includes several so called themes that group together specific workflows or approaches. One of the themes (tobac\_v1) includes the routines from tobac 1.x that are also described in the ACP paper.

Currently included themes:

tobac\_v1



## CHAPTER 4

---

### Analysis

---

tobac provides several analysis functions that allow for the calculation of important quantities based on the tracking results. This includes the calculation of important properties of the tracked objects such as cloud lifetimes, cloud areas/volumes, but also allows for a convenient calculation of statistics for arbitrary fields of the same shape as the input data used for the tracking analysis.



## CHAPTER 5

---

### Plotting

---

tobac provides functions to conveniently visualise the tracking results and analyses.





## CHAPTER 6

---

### Example notebooks

---

tobac is provided with a set of Jupyter notebooks that show examples of the application of tobac for different types of datasets.

The notebooks can be found in a separate repository:

<https://github.com/climate-processes/tobac-tutorials>

The necessary input data for these examples is available on zenodo: [www.zenodo.org/...](http://www.zenodo.org/) and can be downloaded automatically by the Jupyter notebooks.

The examples currently include four different applications of tobac: 1. Tracking of scattered convection based on vertical velocity and condensate mixing ratio for 3D cloud-resolving model output. 2. Tracking of scattered convection based on surface precipitation from the same cloud-resolving model output. 3. Tracking of convective clouds based on outgoing longwave radiation (OLR) for convection-permitting model simulation output. 4. Tracking of convective clouds based on OLR in geostationary satellite retrievals.



### 7.1 Core Modules

---

|   |
|---|
| <code>tobac.analysis.analysis</code>        |
| <code>tobac.analysis.centerofgravity</code> |
| <code>tobac.plot.plotting</code>            |
| <code>tobac.utils</code>                    |

---

### 7.2 Theme Modules: Tobac v1

---

|   |
|---|
| <code>tobac.themes.tobac_v1.</code>             |
| <code>feature_detection</code>                  |
| <code>tobac.themes.tobac_v1.segmentation</code> |
| <code>tobac.themes.tobac_v1.tracking</code>     |
| <code>tobac.themes.tobac_v1.wrapper</code>      |

---