whitebox Documentation

Release 2.3.1

Qiusheng Wu

Apr 05, 2023

Contents:

CHAPTER 1

whitebox-python

1.1 Important Note

This repository is related to the WhiteboxTools Python Frontend only. You can report issues to this repo if you have problems installing this Python package. If you encounter any tool functioning specific errors, please open an issue on Dr. John Lindsay's WhiteboxTools repo.

Links

- Authors: Dr. John Lindsay (https://jblindsay.github.io/ghrg/index.html)
- Contributors: Dr. Qiusheng Wu (https://wetlands.io)
- GitHub repo: https://github.com/opengeos/whitebox-python
- WhiteboxTools: https://github.com/jblindsay/whitebox-tools

- User Manual: https://www.whiteboxgeo.com/manual/wbt_book/intro.html
- PyPI: https://pypi.org/project/whitebox/
- conda-forge: https://anaconda.org/conda-forge/whitebox
- Documentation: https://whitebox.readthedocs.io
- Binder: https://gishub.org/whitebox-cloud
- Free software: MIT license

Contents

1.2 Description

The **whitebox** Python package is built on **WhiteboxTools**, an advanced geospatial data analysis platform developed by Prof. John Lindsay (webpage; jblindsay) at the University of Guelph's Geomorphometry and Hydrogeomatics Research Group. *WhiteboxTools* can be used to perform common geographical information systems (GIS) analysis operations, such as cost-distance analysis, distance buffering, and raster reclassification. Remote sensing and image processing tasks include image enhancement (e.g. panchromatic sharpening, contrast adjustments), image mosaicing, numerous filtering operations, simple classification (k-means), and common image transformations. *WhiteboxTools* also contains advanced tooling for spatial hydrological analysis (e.g. flow-accumulation, watershed delineation, stream network analysis, sink removal), terrain analysis (e.g. common terrain indices such as slope, curvatures, wetness index, hillshading; hypsometric analysis; multi-scale topographic position analysis), and LiDAR data processing. LiDAR point clouds can be interrogated (LidarInfo, LidarHistogram), segmented, tiled and joined, analyized for outliers, interpolated to rasters (DEMs, intensity images), and ground-points can be classified or filtered. *WhiteboxTools* is not a cartographic or spatial data visualization package; instead it is meant to serve as an analytical backend for other data visualization software, mainly GIS.

1.3 Installation

whitebox supports a variety of platforms, including Microsoft Windows, macOS, and Linux operating systems. Note that you will need to have Python 3.x installed. Python 2.x is not supported. The **whitebox** Python package can be installed using the following command:

```
pip install whitebox
```

If you have installed **whitebox** Python package before and want to upgrade to the latest version, you can use the following command:

```
pip install whitebox -U
```

It is recommended that you use a Python virtual environment (e.g., conda) to test the whitebox package. Please follow the conda user guide to install conda if necessary. Once you have conda installed, you can use Terminal or an Anaconda Prompt to create a Python virtual environment. Check managing Python environment for more information.

```
conda create -n wbt python
source activate wbt
conda install whitebox -c conda-forge
```

If you encounter an GLIBC errors when installing the whitebox package, you can try the following command:

```
import whitebox
whitebox.download_wbt(linux_musl=True, reset=True)
```

Alternatively, you can set the environment variable `WBT_LINUX` to `MUSL` before installing the whitebox package. It will automatically download the MUSL version of WhiteboxTools.

```
import os
os.environ["WBT_LINUX"] = "MUSL"
```

1.4 whitebox Tutorials

Launch the whitebox tutorial notebook directly with **mybinder.org** now:

1.4.1 Quick Example

Tool names in the **whitebox** Python package can be called either using the snake_case or CamelCase convention (e.g. *lidar_info* or *LidarInfo*). See below for an example Python script (example.py). If you are interested in using the *WhiteboxTools* command-line program, check WhiteboxTools Usage.

```
import os
import pkg_resources
import whitebox

wbt = whitebox.WhiteboxTools()
print(wbt.version())
print(wbt.help())

# identify the sample data directory of the package
data_dir = os.path.dirname(pkg_resources.resource_filename("whitebox", 'testdata/'))

wbt.set_working_dir(data_dir)
wbt.verbose = False
wbt.feature_preserving_smoothing("DEM.tif", "smoothed.tif", filter=9)
wbt.breach_depressions("smoothed.tif", "breached.tif")
wbt.d_inf_flow_accumulation("breached.tif", "flow_accum.tif")
```

1.4.2 A Jupyter Notebook Tutorial for whitebox

This tutorial can be accessed in three ways:

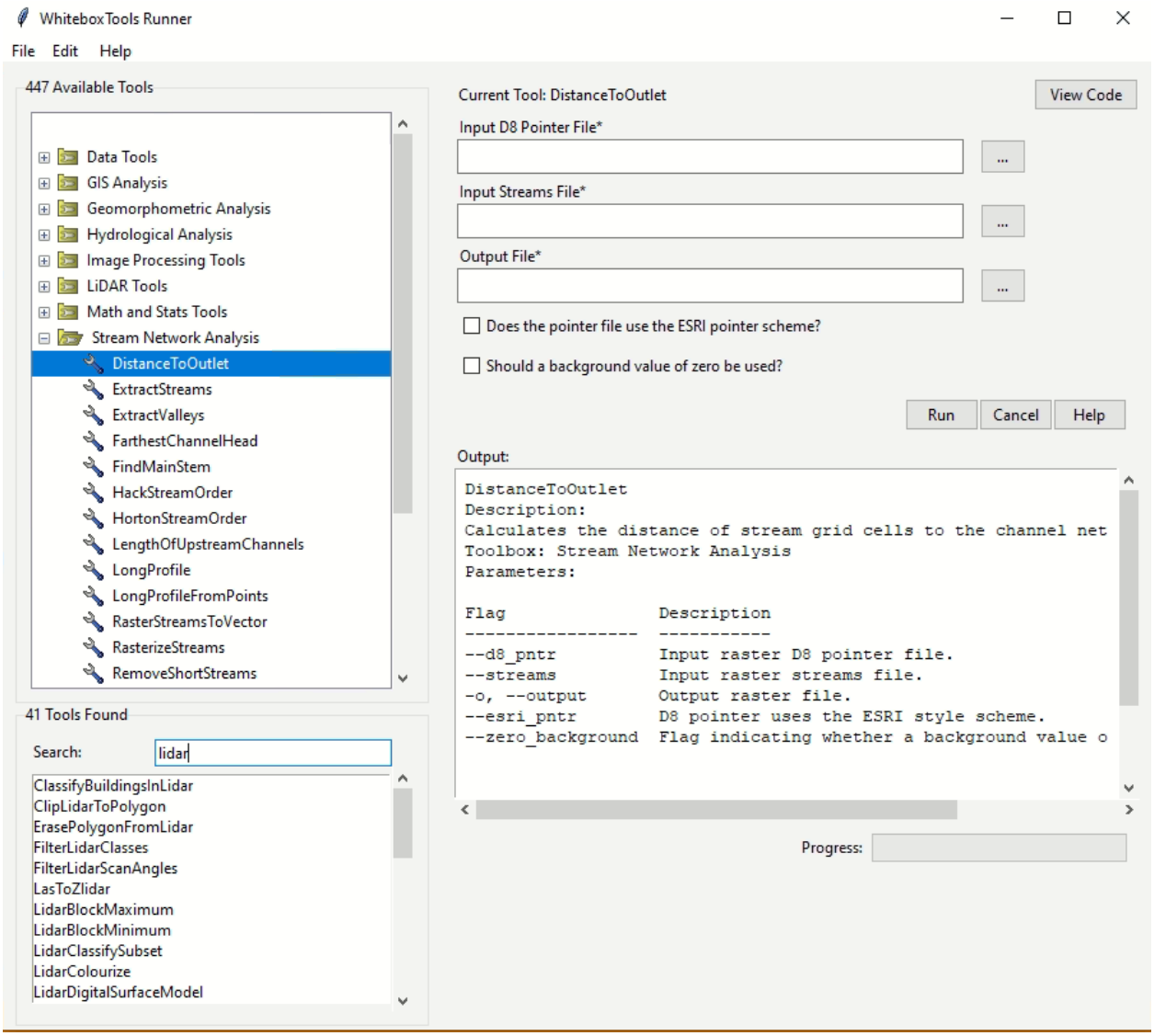
- HTML version: https://gishub.org/whitebox-html
- Viewable Notebook: https://gishub.org/whitebox-notebook
- Interactive Notebook: https://gishub.org/whitebox-cloud

Launch this tutorial as an interactive Jupyter Notebook on the cloud - https://gishub.org/whitebox-cloud.

1.5 whitebox GUI

WhiteboxTools also provides a Graphical User Interface (GUI) - **WhiteboxTools Runner**, which can be invoked using the following Python script:

```
import whitebox
whitebox.Runner()
```



1.6 Troubleshooting

1.6.1 Linux

When using `import whitebox`, if you get an error that says `No module named '_tkinter', please install the python3-tk package`, you can try the following solution:

- For Ubuntu, Linux Mint, etc: `sudo apt-get install python3-tk`
- For Manjaro, Arch Linux: `sudo pacman -S tk`

1.7 Available Tools

The library currently contains **518** tools, which are each grouped based on their main function into one of the following categories: Data Tools, GIS Analysis, Hydrological Analysis, Image Analysis, LiDAR Analysis, Mathematical and

Statistical Analysis, Stream Network Analysis, and Terrain Analysis. For a listing of available tools, complete with documentation and usage details, please see the WhiteboxTools User Manual.

1.8 Supported Data Formats

The WhiteboxTools library currently supports read/writing raster data in Whitebox GAT, GeoTIFF, ESRI (ArcGIS) ASCII and binary (.flt & .hdr), GRASS GIS, Idrisi, SAGA GIS (binary and ASCII), and Surfer 7 data formats. At present, there is limited ability in WhiteboxTools to read vector geospatial data. Support for Shapefile (and other common vector formats) will be enhanced within the library soon.

1.9 Contributing

If you would like to contribute to the project as a developer, follow these instructions to get started:

1. Fork the whitebox project (https://github.com/opengeos/whitebox-python)
2. Create your feature branch (git checkout -b my-new-feature)
3. Commit your changes (git commit -am 'Add some feature')
4. Push to the branch (git push origin my-new-feature)
5. Create a new Pull Request

1.10 License

The **whitebox** package is distributed under the MIT license, a permissive open-source (free software) license.

1.11 Reporting Bugs

Report bugs at https://github.com/opengeos/whitebox-python/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

1.12 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install whitebox, run this command in your terminal:

```
$ pip install whitebox
```

This is the preferred method to install whitebox, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for whitebox can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/opengeos/whitebox
```

Or download the tarball:

```
$ curl  -OL https://github.com/opengeos/whitebox/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use whitebox in a project:

```
import whitebox
```

For example:

```
import os
import pkg_resources
import whitebox

wbt = whitebox.WhiteboxTools()
print(wbt.version())
print(wbt.help())

# identify the sample data directory of the package
data_dir = os.path.dirname(pkg_resources.resource_filename("whitebox", 'testdata/'))

wbt.set_working_dir(data_dir)
wbt.verbose = False
wbt.feature_preserving_denoise("DEM.tif", "smoothed.tif", filter=9)
wbt.breach_depressions("smoothed.tif", "breached.tif")
wbt.d_inf_flow_accumulation("breached.tif", "flow_accum.tif")
```

Check the example.py for more details.

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/opengeos/whitebox/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

whitebox could always use more documentation, whether as part of the official whitebox docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/opengeos/whitebox/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *whitebox* for local development.

1. Fork the *whitebox* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/whitebox.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv whitebox
$ cd whitebox/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 whitebox tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_whitebox
```

4.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 5

Credits

5.1 Development Lead

- Qiusheng Wu <giswqs@gmail.com>

5.2 Contributors

- John Lindsay <jlindsay@uoguelph.ca>

CHAPTER 6

History

6.1 0.2.0 (2018-06-08)

6.2 0.1.0 (2018-06-06)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search