# vegindex

***Release 0.10.2***

**Jul 28, 2022**

# Contents

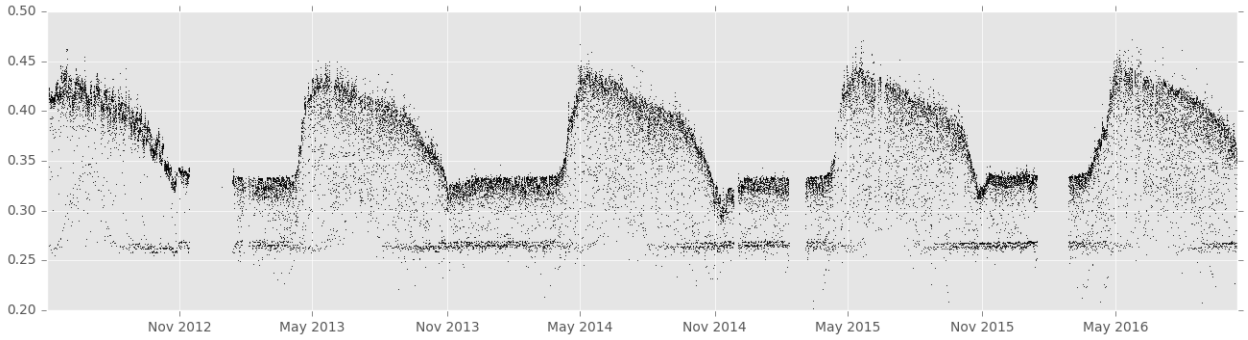



Fig. 1: GCC values from the alligatorriver camera.

# CHAPTER 1

# vegindex

Python tools for generating vegetation index timeseries from PhenoCam images.

- Free software: MIT license

## 1.1 Introduction

The PhenoCam Network is a project designed to study the patterns of seasonal variation (phenology) of vegetation. The project website is at https://phenocam.nau.edu/. The network consists of many cameras collecting images of various types of vegetation. By analysing the images we can quantify the seasonal changes at a particular camera site.

A "vegetation index" refers to a quantity calculated using information from various spectral bands of an image of a vegetated area. The image is typically obtained from a remote-sensing instrument on an earth orbiting satellite. There are several vegetation index values in common usage. The most widely used are NDVI (Normalized Difference Vegetation Index) and EVI (Enhanced Vegetation Index). For the PhenoCam project the Green Chromatic Coordinate or GCC is our standard vegetation index.

For the PhenoCam network, the images are obtained from webcams (usually installed on towers) looking across a vegetated landscape. These images are typically in JPEG format and have 3-bands (Red, Green, and Blue). For some cameras a separate image dominated by an IR (infrared) band is collected.

The algorithms used in in this package have been discussed in numerous publications. You can find a list of publications for the PhenoCam Network project here. The details of the calculation of GCC are presented in this jupyter notebook .

After the package is installed several python scripts should be available as commands:

- `generate_roi_timeseries`

- update_roi_timeseries
- plot_roistats
- generate_summary_timeseries
- generate_roi_ir_timeseries
- update_roi_ir_timeseries
- generate_ndvi_timeseries
- generate_ndvi_summary_timeseries

These scripts allow you to reproduce the PhenoCam network "standard timeseries products" from downloaded data. For a description of the products see the project Tools Page.

## 1.2 Quick Installation

From the command line type:

```
pip install vegindex
```

Some of the packages that vegindex depends on may not install automatically (using pip) since they depend on system libraries. If the above command does not work you can try:

```
pip install Pillow
pip install vegindex
```

The latest version of the documentation can be found at readthedocs.io

CHAPTER 2

# Installation

The current version of the package works with either Python2 or Python 3. The primary use and testing of the package has been done on a Debian linux system. The package had limited testing on OSX and Windows.

## 2.1 Virtual Environments

The `vegindex` package has typically been used in a virtual environment. For Python3 this means having the `venv` package installed. I have also used Anaconda/Miniconda which has it's own virtual environment manager. The use of virtual environments is beyond the scope of this document but using them is highly recommended.

## 2.2 Python `venv` package

When using the `venv` package:

```
mkdir vegindex
cd vegindex
python3 -m venv venv
. venv/bin/activate
```

After that, hopefully, the installation will be as simple as installing the package: into the virtual environment. At the command line:

```
pip install vegindex
```

If you have problems please contact the author or submit a problem report on the **'github page'__**.

:: github page https://github.com/tmilliman/python-vegindex

## 2.3 Using conda/miniconda environments

The steps for using this package in a `conda` environment are

```
conda create --name vegindex python=3.9
conda activate vegindex
conda install numpy matplotlib pillow requests pandas pyephem
pip install vegindex
```

With `pipenv` you can do the following in your working directory:

```
pipenv shell
pipenv install vegindex
```

# CHAPTER 3

# Usage

## 3.1 Command Line Scripts

After installing the `vegindex` package, three python command line scripts will be installed:

- `generate_roi_timeseries`
- `update_roi_timeseries`
- `plot_roistats`
- `generate_summary_timeseries`
- `generate_roi_ir_timeseries`
- `update_roi_ir_timeseries`
- `generate_ndvi_timeseries`
- `generate_ndvi_summary_timeseries`

These scripts allow you to reproduce the PhenoCam network "standard timeseries products" from downloaded data. For a description of the products see the project Tools Page.

## 3.2 Setting up the Data Directory

The vegindex package is designed to work with images downloaded from the PhenoCam network server. To download images you can go to the data tab at the project website.

The images you select come in a zip file with a specific directory structure. For example if we download data from the `harvard` site.

```
harvard
├── 2009
│   ├── 01
```

(continues on next page)

(continued from previous page)

```
│   │   ├── harvard_2009_01_01_110135.jpg
│   │   ├── harvard_2009_01_01_113135.jpg
│   │   ├── harvard_2009_01_01_120135.jpg
 .
 .
 .
│   └── 06
│       ├── harvard_2009_06_01_110139.jpg
│       ├── harvard_2009_06_01_113139.jpg
│       ├── harvard_2009_06_01_120139.jpg
│       └── harvard_2009_06_01_123139.jpg
├── harvard_meta.json
└── harvard_meta.txt
```

where the we have a top level directory for `sitename` then subdirectories for four-digit year and two-digit month, with the image files in the month directories. This is the general structure that this package will expect the image data to be in.

## 3.3 Site-Level Metadata

The package scripts have several ways to get site-level metadata. One level above the site directories you can place a text file (with a default name of `site_info.csv`). If this file is present the scripts will read basic site-level metadata from this file. Here's an example:

```
# This is a site info file
sitename,lat,lon,elev,start_date,end_date,tzoffset,nimage
test,40.00000,-60.00000,300,2008-04-04,2017-11-23,1,300
test2,45.00000,-65.00000,1300,2008-04-04,2017-11-23,1,300
```

If you download images using the link above you will have this information in the included site metadata file. If this file is not present the script try to use the network to get the latest version of this information by pulling information from our server. The pathname of the site-level metadata file can be set using the PHENOCAM_SITE_INFO environment variable.

## 3.4 ROI Lists and Masks

Once you've selected and downloaded the data you would like to process you will need to set up a region of interest (ROI) using an `ROI List` file and the associated `ROI Mask` images.

The `ROI List` file is a simple text file with a list of `ROI mask` images and the dates for which the masks are valid. The `ROI List` format description can be found on this page Here's a simple example where there is only one mask file:

```
#
# ROI List for harvard
#
# Site: harvard
# Veg Type: DB
# ROI ID Number: 0001
# Owner: tmilliman
# Creation Date: 2012-07-12
```

(continues on next page)

(continued from previous page)

```
# Creation Time: 11:42:00
# Update Date: 2014-12-17
# Update Time: 13:55:25
# Description: Deciduous trees in foreground
#
start_date,start_time,end_date,end_time,maskfile,sample_image
2008-04-04,00:00:00,9999-01-01,00:00:00,harvard_DB_0001_01.tif,harvard_2008_04_30_
↪133137.jpg
```

If there are field-of-view shifts you may need additional lines in the list and more mask images. The list file and the mask images need to be placed in a directory named `ROI` under the site name i.e.:

```
harvard
└── ROI
    ├── harvard_DB_0001_01.tif
    └── harvard_DB_0001_roi.csv
```

This file naming convention must also be followed. So the `ROI List` has the form:

```
<sitename>_<ROI-type>_<ROI-sequence-no>_roi.csv
```

and the associated masks are named according to the convention:

```
<sitename>_<ROI-type>_<ROI-sequence-no>_<mask_index>.tif
```

where the "<mask_index>" in the form `nn` is the number in the list of the mask file starting with 01 (e.g. 01, 02, 03, etc.). For the timeseries displayed on the PhenoCam Network website. The `ROI List` files and the `ROI Mask` images are available for download from one of the `ROI Pages` on our site e.g. ROI page for harvard DB_0001

## 3.5 Generating the ROI Image Statistics file

The `generate_roi_timeseries` script reads in the `ROI List` file and `ROI Mask` images. Then for each image found within the timeperiods in the `ROI List` it calculates image statistics over the ROI. You can get help for

```
$ generate_roi_timeseries -h
usage: generate_roi_timeseries [-h] [-v] [-n] site roiname

positional arguments:
site           PhenoCam site name
roiname        ROI name, e.g. DB_0001

optional arguments:
-h, --help     show this help message and exit
-v, --verbose  increase output verbosity
-n, --dry-run  Process data but don't save results
```

The script needs to know where the site images are located. By default it assumes that the site level image directory is at:

```
/data/archive/<sitename>
```

If the images downloaded are in another location, for example `/mydata/directory/harvard`, you can set an an environment variable to specify the path to the images:

```
export PHENOCAM_ARCHIVE_DIR=/mydata/directory/
```

or

```
set PHENOCAM_ARCHIVE_DIR=/mydata/directory/
```

All of the scripts assume the same data layout both for reading and writing.

Here's an example command line session for a bash shell:

```
$ export PHENOCAM_ARCHIVE_DIR=~/Downloads/phenocamdata/
$ generate_roi_timeseries harvard DB_0001
Images processed: 594
Images added to CSV: 594
Total: 594
```

The output format for the "All Image" file can be found here The output CSV file is written to the ROI directory and will follow the name convention: *<sitename>_<vegtype>_<seqno>_roistats.csv*

## 3.6 Generating the 1-day and 3-day Summary Files

The `generate_summary_timeseries` script reads in the "All-Image" file and calculates summary statistics for the 1-day or 3-day period:

```
$ generate_summary_timeseries -h
usage: generate_summary_timeseries [-h] [-v] [-n] [-p [{1,3}]] site roiname

positional arguments:
site                  PhenoCam site name
roiname               ROI name, e.g. canopy_0001

optional arguments:
-h, --help            show this help message and exit
-v, --verbose         increase output verbosity
-n, --dry-run         Process data but don't save results
-p [{1,3}], --aggregation-period [{1,3}]
                      Number of Days to Aggregate (default=1)
```

To generate the 3-day summary file from the "All Image" file generated in the previous section:

```
$ generate_summary_timeseries -p 3 harvard DB_0001
Total: 51
```

A description of the summary files can be found on the project website. The output CSV file is also written to the ROI directory and will follow the name convention: *<sitename>_<vegtype>_<seqno>_[13]day.csv*.

## 3.7 Running `plot_roistats`

The `plot_roistats` python script reads the output of `generate_roi_timeseries` and the 3-day summary generated by `generate_summary_timeseries` script. The `gcc` values for individual images are plotted as points and the three-day 90th-percentile summary is plotted as a line. The color of the plotted points is used to show which data (in red) are filtered before calculating the summary statistics. The default filtering is shown and eliminates

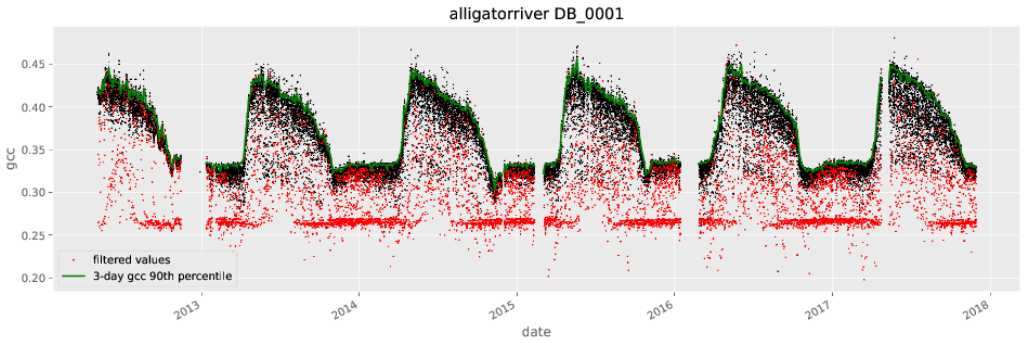
values where the sun elevation angle is low (< 10 degrees) and the mean brightness over the ROI is either low (< 100.) (very dark image) or high ( > 665.) (washed out image).

```
$ plot_roistats -h
usage: plot_roistats [-h] [-v] site roiname

positional arguments:
  site           PhenoCam site name
  roiname        ROI name, e.g. DB_0001

optional arguments:
  -h, --help     show this help message and exit
  -v, --verbose  increase output verbosity
```

The script places the output .pdf file in the ROI directory alongside the .csv files used to produce the plot.




## 3.8 Processing IR Images

Starting with vers 0.10.0 scripts have been added to process the associated IR images. These are used to calculate the 'camera NDVI' timeseries for a given ROI. These scripts rely on having the '.meta' files available to extract the exposure values for both the RGB and IR images. For some sites these '.meta' files are not available.

Generating the camera NDVI time series CSV files involves several steps and understanding the process is helpful:

- generate the RGB ROI timeseries
- generate the IR ROI timeseries
- combine these two files (matching the RGB and IR images) to calculate camera NDVI values for each RGB/IR image pair
- create 1-day and 3-day of the camera NDVI values

## 3.9 Generating the ROI IR Image Statistics File

The `generate_roi_ir_timeseries` script reads in the `ROI List` file and `ROI Mask` images. Then for each IR image found within the timeperiods in the `ROI List` it calculates IR image statistics over the ROI. You can get help for

```
$ generate_roi_ir_timeseries -h

usage: generate_roi_ir_timeseries [-h] [-v] [-n] site roiname

positional arguments:
  site           PhenoCam site name
  roiname        ROI name, e.g. DB_0001

optional arguments:
  -h, --help     show this help message and exit
  -v, --verbose  increase output verbosity
  -n, --dry-run  Process data but don't save results
```

The output CSV file is again written to the ROI directory and will follow the name convention: `<sitename>_<vegtype>_<seqno>_IR_roistats.csv`.

## 3.10 Generating the camera NDVI (RGB/IR Image Pair Statistics) File

The `generate_ndvi_timeseries` script reads in the `RGB roistats` CSV file and `IR roistats` CSV file. Then for each RGB image the script tries to locate the matching IR image. If a match is found then values from the two lines are combined to form a single line with the camera NDVI values.

```
$ generate_ndvi_timeseries --help
usage: generate_ndvi_timeseries [-h] [-v] [-n] site roiname

Merge RGB and IR stats and calculate camera NDVI

positional arguments:
  site           PhenoCam site name
  roiname        ROI name, e.g. canopy_0001

optional arguments:
  -h, --help     show this help message and exit
  -v, --verbose  increase output verbosity
  -n, --dry-run  Process data but don't save results
```

The output file will be written to the ROI directory and will have a name like `<sitename>_<vegtype>_<seqno>_NDVI_roistats.csv`.

## 3.11 Generating the 1-day and 3-day Summary Files

The `generate_ndvi_summary_timeseries` script reads in the "NDVI roistats" file and calculates summary statistics for the 1-day or 3-day period:

```
$ generate_ndvi_summary_timeseries --help

 usage: generate_ndvi_summary_timeseries [-h] [-v] [-n] [-p [{1,3}]]
                                         site roiname

 Generate a summary/aggregated NDVI file

 positional arguments:
```

(continues on next page)

(continued from previous page)

```
  site                  PhenoCam site name
  roiname               ROI name, e.g. canopy_0001

optional arguments:
  -h, --help            show this help message and exit
  -v, --verbose         increase output verbosity
  -n, --dry-run         Process data but don't save results
  -p [{1,3}], --aggregation-period [{1,3}]
                        Number of Days to Aggregate (default=1)
```

The output filename will follow the convention, `<sitename>_<vegtype>_<seqno>_ndvi_[13]day.csv`. TBD

CHAPTER 4

# Reference

## 4.1 vegindex

CHAPTER 5

# Contributing

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

## 5.1 Bug reports

When 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.

## 5.2 Documentation improvements

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

## 5.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/tmilliman/python-vegindex/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 code contributions are welcome :)

## 5.4 Development

To set up *python-vegindex* for local development:

1. Fork python-vegindex (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/python-vegindex.git
```

3. Create a branch for local development:

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

   Now you can make your changes locally.

4. When you're done making changes, run all the checks, doc builder and spell checker with tox one command:

```
tox
```

5. 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
```

6. Submit a pull request through the GitHub website.

### 5.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`)[1].
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

### 5.4.2 Tips

To run a subset of tests:

```
tox -e envname -- py.test -k test_myfeature
```

To run all the test environments in *parallel* (you need to `pip install detox`):

```
detox
```

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.
It will be slower though ...

CHAPTER 6

# Authors

- Thomas Milliman - https://github.com/tmilliman

# CHAPTER 7

# Changelog

0.10.2 (2022-07-27) * Fix bug in update_summary_timeseries * change references to UNH to NAU

## 7.1 0.10.0 (2021-11-30)

- Add 3.9 tox environment to tox.ini
- Add IR and camera NDVI processing scripts
- Drop support for python2

## 7.2 0.7.2 (2020-04-12)

- Add 3.8 tox environment to tox.ini

## 7.3 0.7.0 (2020-04-01)

- Add checks when reading an ROI List CSV file that the time ranges are in order and don't overlap

## 7.4 0.6.1 (2019-07-15)

- Update installation doc.
- Simple test for reading roistats file.

## 7.5 0.6.0 (2019-07-12)

- Add awbflag (auto white-balance) to roistats file

## 7.6 0.5.3 (2019-04-03)

- Update requirements for python 3.7
- Add prefix to regular expressions

## 7.7 0.5.2 (2018-04-09)

- Really fix bug in plot_roistats when no data are filtered.

## 7.8 0.5.1 (2018-04-09)

- Fix bug in plot_roistats when no data are filtered.
- Update docs

## 7.9 0.5.0 (2017-11-29)

- Fix header on roistats.csv file
- Add plotting script (matplotlib library is now required)
- Remove timeout on requests query which was causing tests to fail.
- Update usage docs

## 7.10 0.4.0 (2017-11-27)

- Add fallback to local site_info.csv file to get basic site metadata
- Update exception handling (removed bare exceptions)

## 7.11 0.3.1 (2017-10-06)

- Change data product name from _roi_statistics.csv to _roistats.csv

## 7.12 0.3.0 (2017-09-12)

- Added support for .meta files
- Change data product name from _timeseries.csv to _roi_statistics.csv

## 7.13 0.2.0rc1 (2017-06-14)

- Added support for python3

## 7.14 0.1.1rc3 (2017-06-13)

- First release on PyPI.

CHAPTER 8

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## v

# Index

## V