

sensortoolkit beta-testing instructions

Samuel Frederick, ORAU Contractor for U.S. EPA ORD

Office: 919-541-4086 | Email: frederick.samuel@epa.gov

Last updated 9/16/2021, sensortoolkit v0.4.1 beta 2

Thank you for participating in the internal beta-testing of sensortoolkit! Your feedback in running through the library features will be valuable in improving sensortoolkit for public release.

What is sensortoolkit?

sensortoolkit is a Python library for evaluating air sensor data. The library is intended for use with sensors collocated at ambient air monitoring sites alongside FRM/FEM monitors for comparison and analysis of sensor data against reference-grade data.

How can sensortoolkit help users evaluate sensor data?

- Import sensor data via a standardized ingestion process and interactive setup module
- Conduct analysis with the `SensorEvaluation` module
 - Average to 1-hour and/or 24-hour averaging intervals.
 - Import FRM/FEM reference data from a variety of sources, including ingestion modules for importing data from AirNowTech, and modules for querying either the AQS or AirNow API services.
 - Submit queries for single or multiple parameters, parse datasets into a consistent reference data format and save unmodified and processed datasets to a data directory.
 - Compute U.S. EPA's recommended performance metrics for evaluating PM_{2.5} and O₃ sensors.
 - Visualize sensor performance with various figures and save to file location.
 - Sensor vs. FRM/FEM scatter plots
 - Timeseries indicating both sensor and FRM/FEM concentrations
 - Performance metric results and comparison against target values/ranges.
 - Save performance evaluation results, statistics, and supplemental information detailing the deployment conditions to a deployment JSON file.
- Create testing reports using U.S. EPA's base-testing report template (PowerPoint file) with the `PerformanceReport` module.
- Additional modules are included for calculating quantities (AQI, PM2.5 NowCast, application of sensor correction equations, the U.S. Wide correction equation for PurpleAir sensors via Barkjohn et al. 2021, etc.) and conducting additional analysis (quality control modules for identifying outliers, invalidation of datapoints, A&B channel cleaning for PurpleAir data via Barkjohn et al. 2021, etc.)
- Access to modules utilized by the `SensorEvaluation` and `PerformanceReport` for greater customization in conducting analysis.
- HTML-based documentation is provided alongside the library to guide individuals through library use and API reference for working with individual modules.

Who is the intended audience for this library?

sensortoolkit is most suitable for individuals who have some prior coding experience in python. The library is equipped with an API (application programming interface) that allows for ease of navigation and customization, making sensortoolkit accessible to individuals with a wide range of skillsets (e.g., individuals interested in monitoring their own sensor data, students and academic researchers, and industry professionals).

Part 1: Installing and Updating sensortoolkit

Prerequisites

- Prior to the steps listed below, individuals interested in beta-testing will be granted access to the BitBucket repository where the library code is maintained. BitBucket is used by EPA for hosting private, internal projects and for maintaining version control for projects intended for public release that have not undergone clearance yet.

****Please make sure you are able to view the repository at the following link:**

<https://bitbucket.epa.gov/users/sfrede01/repos/sensortoolkit/browse>******

If you are not able to view the repository, please reach out to me (frederick.samuel@epa.gov) and I will grant you access to the repository.

- Beta-testers will need the following software installed to run sensortoolkit:
 - [git](#) (on the list of preapproved software, email EISD directly to have git installed on your system). git is a version control system that streamlines the process of downloading (cloning) online repositories, updating local copies, and suggesting changes to the online repository based on modifications to your local repository.
 - A python (>=v3.7) distribution (I highly recommend [Anaconda](#) since it comes with the Spyder integrated development environment (IDE), and is what I have used to develop sensortoolkit. Users will need to submit a software request to have Anaconda installed).
 - A command line interface (CLI) for submitting various git commands and installing/updating the sensortoolkit library. I recommend using Anaconda Prompt, which comes alongside an Anaconda installation. I recommend this prompt over the Windows Command Prompt, mainly because users may occasionally need to install or update python packages, and for Anaconda installations, this is handled by the [conda package manager](#). Conda is built in to the Anaconda Prompt, so using this CLI is preferred.

Cloning the repository

1. Open up Anaconda Prompt and navigate to a directory where you would like to install the repository. This directory will be the parent directory of the repository. For example, if you navigate to the `/Documents` directory via the CLI prompt:

```
cd C:/Users/.../Profile/Documents
```

you should expect the repository to be located at the following path: `C:/Users/.../Profile/Documents/sensortoolkit`

2. Clone the repository with git using the following CLI prompt:

```
git clone https://bitbucket.epa.gov/scm/~sfrede01/sensortoolkit.git
```

This will download the most recent version of the repository on BitBucket to a new folder `/sensortoolkit` within the user's current working directory.

Installing the library with pip

1. Once you've cloned the repository, navigate to the folder location for the cloned repository. If you're continuing from the instructions above for cloning the repository, you should be able to use the following CLI prompt to change directories:

```
cd sensortoolkit
```

2. Now that you're in the library folder, the `sensortoolkit` needs to be installed to a target directory where python looks for packages whenever the user tells python to import a package name. By default, this is the `/site-packages` directory, and should be located at a path that looks something like `C:\Users\your_LAN_ID\Anaconda3\Lib\site-packages` (if you have Anaconda installed). The location of this package may be a little different depending on how your python installation was configured, although this shouldn't matter too much.

Type the following CLI prompt to install `sensortoolkit` (don't forget the period!):

```
pip install .
```

The installation process checks for a number of packages `sensortoolkit` needs to run (dependencies).

If you have Anaconda installed, you'll notice that the installation process may indicate that a lot of the required libraries are already installed as those packages come with the base installation of Anaconda. Pip may need to install `python-pptx`, which is used by `sensortoolkit` to create testing reports as `.pptx` files.

Importing the `sensortoolkit` Library

1. Once you've downloaded and installed the library, you may want to ensure that `sensortoolkit` has been installed correctly. Still within the Anaconda Prompt, type the following CLI prompt:

```
python
```

This will launch python within the Anaconda Prompt in interactive mode. Next, type the following python commands (*Note: If you're copying and pasting commands in, you will need to copy one line at a time*):

```
import sensortoolkit
sensortoolkit.__version__
```

This should print the version of the library that you downloaded from BitBucket and indicates that the library has been properly installed.

Tip: You can exit the interactive python console by typing `exit()`.

2. Up till now, we've been using the Anaconda Prompt (or an equivalent CLI) for download, installing, and importing the `sensortoolkit` library. For making the best use of the library and modules, I highly recommend beta-testers use an IDE such as Spyder that comes installed with an Anaconda distribution. Spyder (much like R-Studio) provides users with a platform for scripting and executing code as well as useful utilities such as the variable explorer that can be a valuable tool when working with sensor datasets.

In order to open Spyder, you can either type into the Windows Search Menu 'spyder' (you should see an icon come up that says 'Spyder (Anaconda3)'), or if you still have the Anaconda Prompt open, you can enter the prompt `spyder`.

Updating `sensortoolkit` for beta-testers

1. Open the Anaconda Prompt and navigate to the location where you downloaded the repository:

```
cd path/to/sensortoolkit
```

2. Check for updates via the following git command. If your version of the library is outdated, relevant updates will be downloaded, but won't be incorporated into your local library yet.

```
git fetch
```

The next command incorporates the changes retrieved via the `fetch` command:

```
git merge
```

Note: If you know your local library is out of date and agree to the changes to update the library, these two git commands can be combined into one `git pull` command.

3. Next, you will need to update the installation of sensortoolkit in your target directory. This is done via the following CLI prompt:

```
pip install --upgrade .
```

Resources

Below is the file structure for the sensortoolkit library (this assumes the library has been downloaded in the Documents folder):

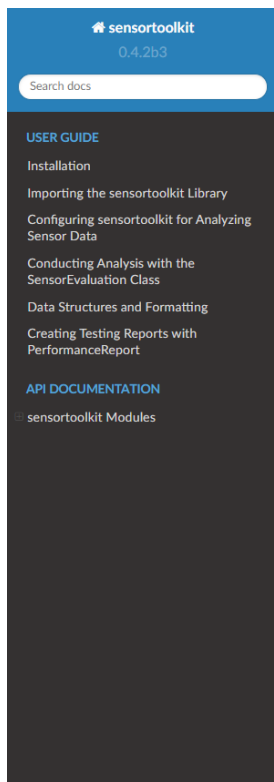
```
Documents
├──sensortoolkit
│   ├──docs
│   ├──beta_testing
│   │   ├──data
│   │   └──example_scripts
│   └──sensortoolkit
```

<- Your cloned copy of the library
<- HTML documentation
<- Items for beta testers
<- Example sensor dataset for testing
<- Example scripts for setting up an analysis, conducting analysis, and creating
<- Library source code

Documentation

The `/sensortoolkit/docs` folder contains the HTML documentation for the library. This documentation is intended for end users once the library is released to the public, so some of the installation discussion may differ from what is covered in this beta-testing guide. However, the HTML documentation is intended to provide a comprehensive overview of the sensortoolkit library and its use, so I would recommend beta testers review the documentation.

- In order to view the documentation, go to `sensortoolkit/docs/build/html`. Double click to open the file named `index.html`, which is the main landing page for the documentation. The documentation should open in a browser, and should look like the following landing page:



sensortoolkit - Air Sensor Data Analysis Library



Important

This documentation is currently under development. Please check for updates to the BitBucket repository before referencing the documentation.

sensortoolkit is a Python library for evaluating air sensor data. The library is intended for use with sensors collocated at ambient air monitoring sites alongside FRM/FEM monitors for comparison and analysis of sensor data against reference-grade data.

sensortoolkit can be used to evaluate sensor data for a single or multiple sensors measuring one of the following pollutants: PM_{10} , $PM_{2.5}$, PM_{10} , CO , CO_2 , NO , NO_2 , NO_x , O_3 , SO_2 , SO_x .

U.S. EPA's Performance Targets Reports

In February 2021, U.S. EPA released two reports outlining recommended performance testing protocols, metrics, and target values for fine particulate matter ($PM_{2.5}$) and ozone (O_3) air sensors used in non-regulatory, supplemental, and informational monitoring (NSIM) applications.

Beta-testing: Example Scripts and Datasets

The folder `/sensortoolkit/beta_testing` contains a number of example scripts and datasets that we will use later on during component testing. We will return to these items later on during component testing.

Part 2: Component Testing

1. Within the `/Documents` folder, create a new folder named `sensortoolkit_testing`.
2. Open Spyder and create a new file (*File* → *New File*).
3. Save this file as `component_testing.py` (*File* → *Save As*) and save the file within the `Documents/sensortoolkit_testing` folder. This is the script you will use to test out the features of the *sensortoolkit* library.

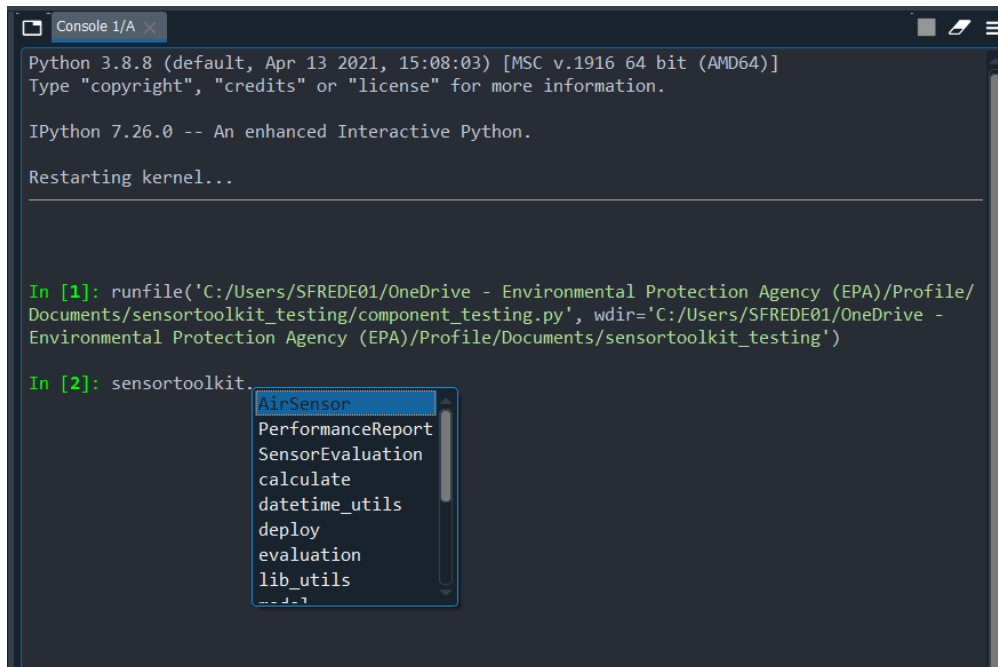
The directory structure in your Documents folder should now look something like this:

```
Documents
├── sensortoolkit                    <- Your cloned copy of the library
│   ├── docs
│   ├── beta_testing
│   │   ├── data
│   │   └── example_scripts
│   └── sensortoolkit
├── sensortoolkit_testing           <- Your testing directory
└── component_testing.py
```

4. Within `component_testing.py`, import the library by typing the following command:

```
import sensortoolkit
```

5. Run the file by pressing the green arrow in the upper left corner (or press F5). In the console, you should see a runfile statement that indicates the `component_testing.py` is being executed.
6. Once the file has run, type into the the console `sensortoolkit` followed by a period `.`. Press Tab and you will be able to view the various components in the sensortoolkit library (see below). This dropdown includes objects, sub-packages, and methods for working with sensor data. We'll focus on the objects first (items in upper-case within the drop down menu, which coincidentally are also listed at the top).



```
Python 3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.26.0 -- An enhanced Interactive Python.

Restarting kernel...

In [1]: runfile('C:/Users/SFREDE01/OneDrive - Environmental Protection Agency (EPA)/Profile/
Documents/sensortoolkit_testing/component_testing.py', wdir='C:/Users/SFREDE01/OneDrive -
Environmental Protection Agency (EPA)/Profile/Documents/sensortoolkit_testing')

In [2]: sensortoolkit.
```

The dropdown menu shows the following items: AirSensor, PerformanceReport, SensorEvaluation, calculate, datetime_utils, deploy, evaluation, lib_utils, and ... and more.

sensortoolkit objects

sensortoolkit contains a number of class objects to aid users in various tasks, including working with sensor data, keeping track of attributes pertaining to the sensor(s) and parameter(s) being evaluated, conducting analysis, and producing reports.

Since users may wish to evaluate multiple sensor types and numerous pollutants, our goal in organizing the evaluation workflow into a set of recognizable and consistent class objects is to provide an easy-to-navigate platform for conducting evaluations and data analysis, while also allowing a high degree of customization. We hope that as you walk through the component testing, you will determine whether the library and its organization meets these goals. We also hope you may choose to comment on any suggestions or recommendations to meet these goals for ease of use by responding to the charge questions in Part 3.

sensortoolkit.AirSensor

The first object the user interacts with is the `AirSensor` object. As the name suggests, the `AirSensor` object is used to organize information about the sensor being evaluated, including the make and model, a list of parameters the user intends to evaluate for the sensor, serial identifiers indicating the unique sensor units being tested, and information about the data ingestion configuration.

In `component_testing.py`,

AirSensor Methods

Name	Description	Arguments
<code>AirSensor.create_directories()</code>	Create the folder structure for storing sensor and reference data, figures, evaluation statistics, and reports.	None

Name	Description	Arguments
<code>AirSensor.copy_datasets()</code>	Copy sensor datasets from specified folder location into the appropriate <code>/Data and Figures/sensor_data</code> subdirectory.	None
<code>AirSensor.sensor_setup()</code>	Interactive module for configuring the sensor data ingestion process. The user is guided through a number of prompts to construct the appropriate ingestion configuration.	None

`sensortoolkit.Parameter`

`sensortoolkit.SensorEvaluation`

`sensortoolkit.PerformanceReport`

sensortoolkit methods

sensortoolkit's objects reference numerous methods (i.e., standalone functions), that the user can interact with through the library's API.

Part 3: Charge Questions

Following component testing, please respond to the following questions regarding the sensortoolkit library.