Museo ToolBox : a python library for remote sensing including a new way to handle rasters.

13 December 2019

# Summary

Museo ToolBox is a python library dedicated to the processing of images in remote sensing.

In this domain, classifying land cover type is a common and sometimes complex task, regardless of your level of expertise. Recurring procedures such as extracting Regions Of Interest (raster values from your polygon), computing spectral indices or validate a model with a cross-validation can be difficult to implement.

Museo ToolBox aims to simplify the whole process by making the main treatments more accessible (extracting of Region Of Interests, fitting a model by using a cross-validation, computing NDVI or various spectral indices, performing any kind of array function to your raster)…

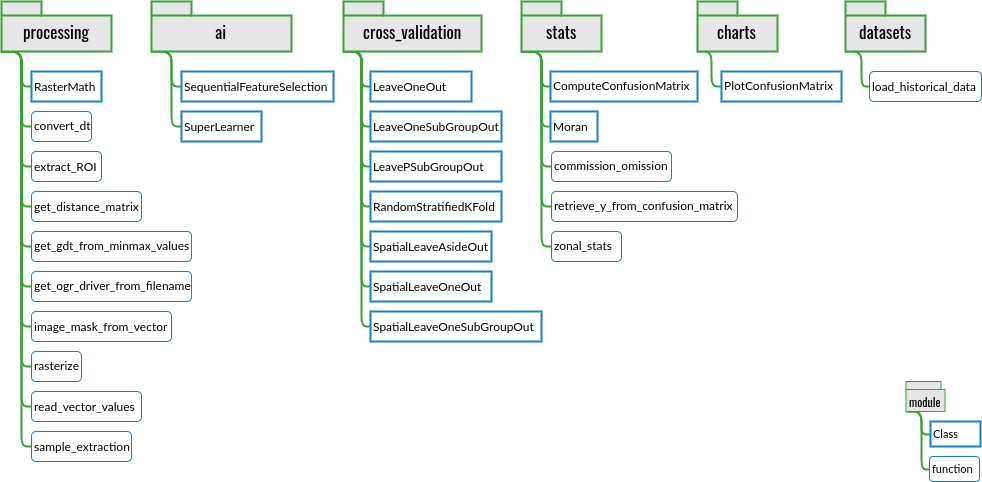
The main objective of this library is to ease the transposition of array-like functions into to an image and to promote good practices in machine learning.

To make Museo ToolBox easier to get started with, a [full documentation with lot of examples is available online on read the docs](http://museotoolbox.readthedocs.io/).

# Museo ToolBox in details

Museo ToolBox is organized into several modules (Figure 1) :

* [processing](https://museotoolbox.readthedocs.io/en/latest/modules/museotoolbox.processing.html) : raster and vector processing.
* [cross-validation](https://museotoolbox.readthedocs.io/en/latest/modules/museotoolbox.cross_validation.html) : stratified cross-validation compatible with scikit-learn.
* [ai](https://museotoolbox.readthedocs.io/en/latest/modules/museotoolbox.ai.html) : artificial intelligence module built upon scikit-learn.
* [charts](https://museotoolbox.readthedocs.io/en/latest/modules/museotoolbox.charts.html) : plot confusion matrix with F1 score or producer/user’s accuracy.
* [stats](https://museotoolbox.readthedocs.io/en/latest/modules/museotoolbox.stats.html) : compute stats (like Moran’s Index, confusion matrix, commision/omission) or extract truth and predicted label from a confusion matrix.



Museo ToolBox schema.

Here are some main usages of Museo ToolBox :

1. [Read and write a raster block per block using your own function](https://museotoolbox.readthedocs.io/en/latest/modules/processing/museotoolbox.processing.RasterMath.html).
2. [Generate a cross-validation, including spatial cross-validation](https://museotoolbox.readthedocs.io/en/latest/auto_examples/index.html#cross-validation).
3. [Fit models with scikit-learn, extract accuracy from each cross-validation fold, and predict raster](https://museotoolbox.readthedocs.io/en/latest/modules/ai/museotoolbox.ai.SuperLearner.html).
4. [Plot confusion matrix and add f1 score or producer/user accuracy](https://museotoolbox.readthedocs.io/en/latest/modules/charts/museotoolbox.charts.PlotConfusionMatrix.html#museotoolbox.charts.PlotConfusionMatrix).
5. [Get the y\_true and and y\_predicted labels from a confusion matrix](https://museotoolbox.readthedocs.io/en/latest/modules/stats/museotoolbox.stats.retrieve_y_from_confusion_matrix.html).

## RasterMath

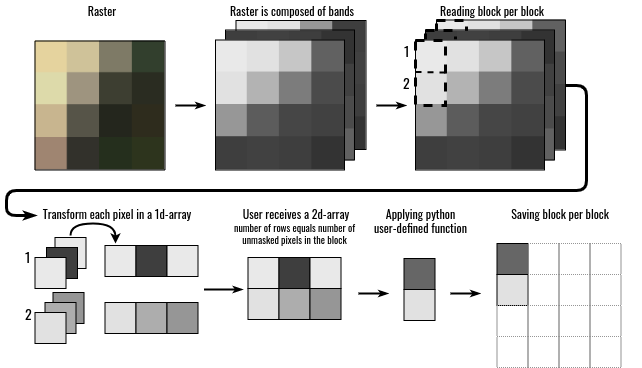
Available in museotoolbox.processing, RasterMath class is the keystone of Museo ToolBox.

The question I asked myself is : How can we make it as easy as possible to implement array-like functions to images ? The idea behind RasterMath is, If the function is intended to operate with an array, it should be easy to use it with your raster using as few lines as possible.

So, what does RasterMath really do ? The user only works with an array an confirms with a sample that the process is doing well, and let RasterMath generalizing to the whole image. The user doesn’t have to manage the raster reading and writing process, the no-data management, the compression, the number of bands or the projection. Figure 2 explains how RasterMath read a raster, performs your function, and write it to a new raster.

The objective is to **let the user only focus on his array-compatible function**, andRasterMath manages the raster part.

[See RasterMath documentation and examples](https://museotoolbox.readthedocs.io/en/latest/modules/processing/museotoolbox.processing.RasterMath.html)



RasterMath under the hood

## Artificial Intelligence

The artificial intelligence (ai) module is natively built to implement scikit-learn algorithm and uses state of the art methods (such as standardizing the input data). SuperLearner class optimizes the fit process by a grid search to fix parameters of the classifier. There is also a Sequential Feature Selection protocol which supports number of components (e.g. a single-date image is composed of four bands, i.e. 4 features, so you may select 4 features at once).

[See the SuperLearner documentation and examples](https://museotoolbox.readthedocs.io/en/latest/modules/ai/museotoolbox.ai.SuperLearner.html)

## Cross-validation

Museo ToolBox produces only stratified cross-validation, which means the separation between the training and the validation samples is made by respecting the size per class. For example the Leave-One-Out method will keep one sample of validation per class. As stated by [@olofsson\_good\_2014] *“stratified random sampling is a practical design that satisfies the basic accuracy assessment objectives and most of the desirable design criteria”*. For spatial cross-validation, see [@karasiak\_2019] inspired from [@roberts\_2017].

Museo ToolBox offers two different kind of cross-validation :

### Non-spatial cross-validation

* Leave-One-Out.
* Leave-One-SubGroup-Out.
* Leave-P-SubGroup-Out (Percentage of subgroup per class).
* Random Stratified K-Fold.

### Spatial cross-validation

* Spatial Leave-One-Out [@karasiak\_2019].
* Spatial Leave-Aside-Out.
* Spatial Leave-One-SubGroup-Out (using centroids to select one subgroup and remove other subgroups for the same class inside a specified distance buffer).

[See the cross-validation documentation and examples](https://museotoolbox.readthedocs.io/en/latest/auto_examples/index.html#cross-validation)

# Acknowledgements

I acknowledge contributions from [Mathieu Fauvel](http://fauvel.mathieu.free.fr/), beta-testers (hey Yousra Hamrouni !), my thesis advisors : Jean-François Dejoux, Claude Monteil and [David Sheeren](https://dsheeren.github.io/). Many thanks to Marie for proofreading.

# References