InVEST Scripting Framework Guide

Since version 2.5.5, InVEST supports the execution of custom Python scripts to call against the core InVEST API outside of the user interface. This functionality is useful for batching InVEST runs, model parameter studies, and/or creating new ecosystem service models built from existing InVEST models. This guide provides information about installing the scripting environment, generating and executing scripts in that environment, as well as examples of common applications.

Installation of the InVEST Scripting/Development Environment

The following is a step by step guide to install the InVEST 2.5.5 development environment for scripting.

- First download and install Python 2.7 (32-bit) (64-bit)
- Next, download and install the following Python packages and install them into the Python 2.7 installation from the first step. All of these packages are 3rd party open source dependencies of InVEST 2.5.5.
    - Cython v1.7.1 or later (Download)
    - Numpy Superpack v.0.11.0 (Download)
    - GDAL v1.9.0 or later (Download)
    - PIL (Download)
    - py2exe (Download)
    - PyQt v4.7 or later(Download)
    - Scipy Superpack v0.11.0 (Download)
    - Setuptools (Download)
    - pip (Download)
    - Nose v1.2.1 or later (Download)
    - Shapely (Download)
    - PyAMG v.2.0.5dev (Download)
    - Virtualenv v1.8.0 or later (Download)
    - Poster v0.8.1 or later (install with pip by typing the following into a windows shell window after the pip package is installed above `C:\Python27\Scripts\pip.exe install poster`)
    - Matplotlib (Download)
- Finally, download and install the "InVEST Python extensions" from http://www.naturalcapitalproject.org/download.html.

Writing an InVEST Python Script

This section provides guidance about how to generate an InVEST Python script, how to execute it in the InVEST Python environment, and how to modify that script for custom user needs. It assumes the user is familiar with using the traditional standalone InVEST toolset and that at

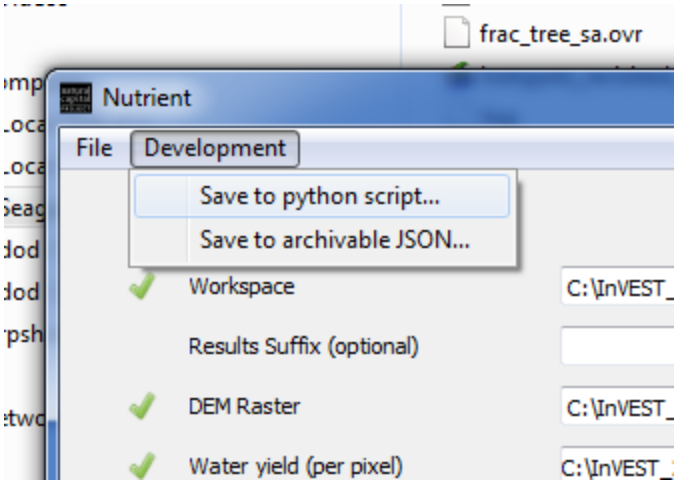
InVEST 2.5.6 or later is installed on the user's computer.

1. **Launch InVEST Model:** Once an InVEST model is selected for scripting, launch that model from the Windows Start menu. This example in this guide follows the Nutrient Retention model.

2. **Fill in InVEST Model Input Parameters:** Once the user interface loads, populate the inputs in the model likely to be used in the Python script. For testing purposes the default InVEST's data is appropriate. However, if a user wishes to write a batch several InVEST runs, it would be reasonable to populate the user interface with data for the first run.

3. **Generate a sample Python Script from the User Interface:** Open the Development menu at the top of the user interface and select "Save to python script..." and save the file to a known location.

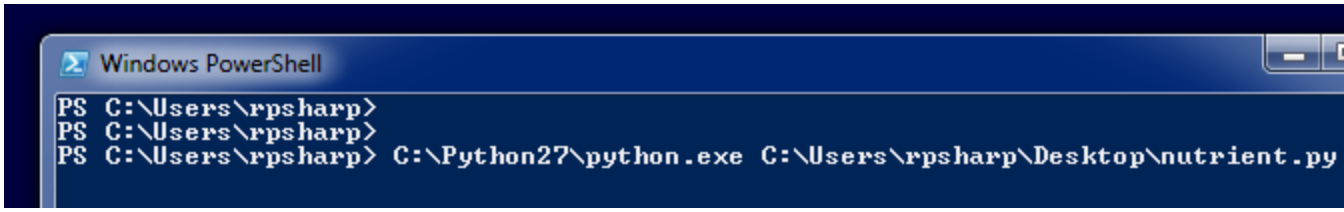
   (screenshot)



4. **Execute the script in the InVEST Python Environment:** Launch a Windows PowerShell from the Start menu (type "powershell" in the search box), then invoke the Python interpreter on the InVEST Python script from that shell. In this example the Python interpreter is installed in `C:\Python27\python.exe` and the script was saved in `C:\Users\rpsharp\Desktop\nutrient.py`, thus the command to invoke the interpreter is:

   ```
   > C:\Python27\python.exe C:\Users\rpsharp\Desktop\nutrient.py
   ```

   (screenshot)



5. **Output Results:** As the model executes, status information will be printed to the console. Once complete, model results can be found in the workspace folder selected during the initial configuration.

Guidance for modifying an InVEST Python Script

InVEST Python scripts consist of two sections:

- The argument dictionary that represents the model's user interface input boxes and parameters.
- The call to the InVEST model itself.

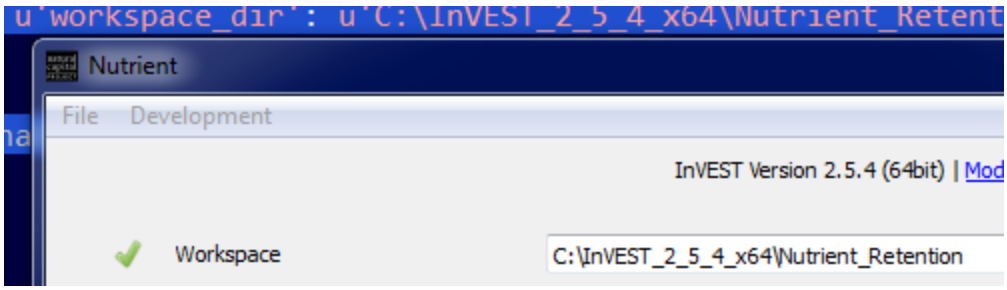
For reference, consider the following script generated by the Nutrient model with default data inputs:

```
"""
This is a saved model run from invest_natcap.nutrient.nutrient.
Generated: 07/17/13 14:28:41
InVEST version: 2.5.4
"""

import invest_natcap.nutrient.nutrient

args = {
        u'accum_threshold': u'1000',
        u'biophysical_table_uri':
u'C:\InVEST_2_5_4_x64\WP_Nutrient_Retention\Input\water_biophysical_t
able.csv',
        u'calc_n': True,
        u'calc_p': True,
        u'suffix': '',
        u'dem_uri': u'C:\InVEST_2_5_3\Base_Data\Freshwater\dem',
        u'landuse_uri':
u'C:\InVEST_2_5_3\Base_Data\Freshwater\landuse_90',
        u'pixel_yield_uri':
u'C:\InVEST_2_5_4_x64\WP_Nutrient_Retention\Input\wyield.tif',
        u'valuation_enabled': True,
        u'water_purification_threshold_table_uri':
```

```
u'C:\InVEST_2_5_4_x64\WP_Nutrient_Retention\Input\water_purification_
threshold.csv',
        u'water_purification_valuation_table_uri':
u'C:\InVEST_2_5_4_x64\WP_Nutrient_Retention\Input\water_purification_
valuation.csv',
        u'watersheds_uri':
u'C:\InVEST_2_5_3\Base_Data\Freshwater\watersheds.shp',
        u'workspace_dir': u'C:\InVEST_2_5_4_x64\Nutrient_Retention',
}

invest_natcap.nutrient.nutrient.execute(args)
```

Elements to note:

- **Parameter Python Dictionary:** Key elements include the `'args'` dictionary. Note the similarities between the key values such as `'workspace_dir'` and the equivalent "Workspace" input parameter in the user interface. Every key in the `'args'` dictionary has a corresponding reference in the user interface.



  In the example below we'll modify the script to execute the nutrient model for a parameter study of `'accum_threshold'`.

- **Execution of the InVEST model**: The InVEST API invokes models with a consistent syntax where the module name that contains the InVEST model is listed first and is followed by a function called `'execute'` that takes a single parameter called `'args'`. This parameter is the dictionary of input parameters discussed above. In this example, the line

  ```
  invest_natcap.nutrient.nutrient.execute(args)
  ```

  executes the nutrient model end-to-end. If the user wishes to make batch calls to InVEST, this line will likely be placed inside a loop.

Example: Accumulation Threshold Parameter Study

This example executes the InVEST Nutrient model on 10 values of accumulation threshold stepping from 500 to 1000 pixels in steps of 50. To modify the script above, replace the execution call with the following loop:

```
#Loops through the values 500, 550, 600, ... 1000
for accum_threshold in range(500, 1001, 50):
     #set the accumulation threshold to the current value in the
loop
     args['accum_threshold'] = accum_threshold
     #set the suffix to be accum### for the current accum_threshold
     args['suffix'] = 'accum' + str(accum_threshold)
     invest_natcap.nutrient.nutrient.execute(args)
```

This loop executes the InVEST nutrient model 10 times for accumulation values 500, 550, 600, … 1000 and adds a suffix to the output files so results can be distinguished.

Example: Invoke Nutrient Model on a directory of Land Cover Maps

In this case we invoke the InVEST nutrient model on a directory of land cover data located at `C:\User\Rich\Desktop\landcover_data`. As in the previous example, replace the last line in the UI generated Python script with:

```
import os
landcover_dir = u'C:\User\Rich\Desktop\landcover_data'
#Loop over all the filenames in the landcover dir
for landcover_file in os.listdir(landcover_dir):
     #Point the landuse uri parameter at the directory+filename
     args['landuse_uri'] =
os.path.join(landcover_dir,landcover_file)
     #Make a useful suffix so we can differentiate the results
     args['suffix'] = 'landmap' +
os.path.splitext(landcover_file)[0]
     #call the nutrient model
     invest_natcap.nutrient.nutrient.execute(args)
```

This loop covers all the files located in `C:\User\Rich\Desktop\landcover_data` and updates the relevant `'landuse_uri'` key in the args dictionary to each of those files during execution as well as making a useful suffix so output files can be distinguished from each other.

Summary

The InVEST scripting framework was designed to assist InVEST users in automating batch runs or adding custom functionality to the existing InVEST software suite. Support questions can be

directed to the NatCap support forums at http://ncp-yamato.stanford.edu/natcapforums/.