

Getting to know IN-CORE 2.0

Last change 07/17/19

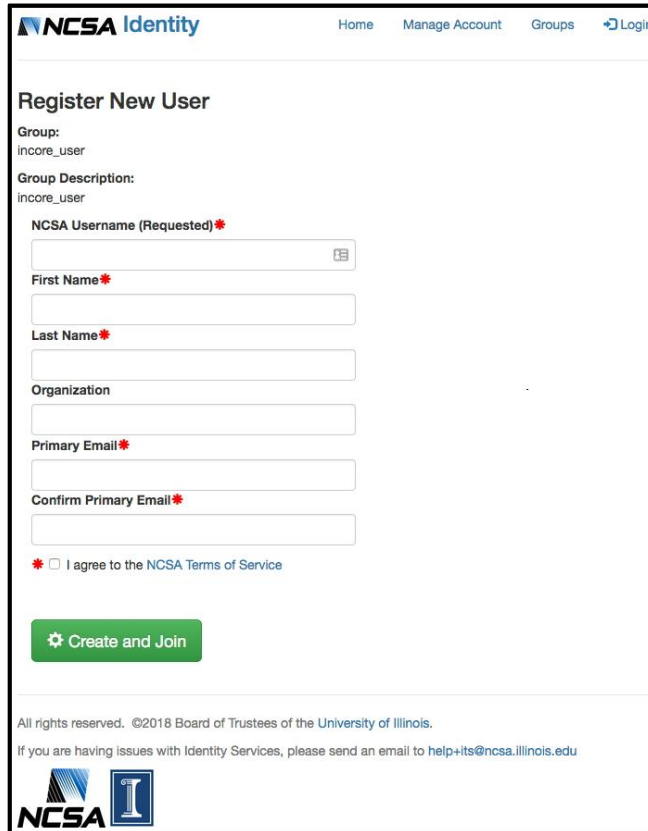
NCSA Team, incore-dev@lists.illinois.edu

Table of Contents

1	Acquiring IN-CORE account	2
2	Installing pyIncore	3
2.1	Prerequisites	3
2.2	Windows 64-bit	4
2.3	Mac OS	6
2.4	Linux	7
3	Testing pyIncore Installation	8
3.1	Running a Building Damage Analysis Locally	8
4	Using IN-CORE Lab	11
4.1	Running Jupyter Notebook in IN-CORE Lab	11
5	How to Contact and Work with NCSA	13
6	Information for pyIncore Developers	13
7	Additional Information	14
7.1	Technical Documentation	14
7.2	IN-CORE Web tools.....	15
7.3	Additional Information about IN-CORE Lab.....	17

1 Acquiring IN-CORE account

- A user must have a valid IN-CORE account recognized by the IN-CORE service. Please **register** at <https://identity.ncsa.illinois.edu/register/UUMK36FU2M>



The screenshot shows the 'Register New User' page on the NCSA Identity website. The page has a header with the NCSA Identity logo and navigation links for Home, Manage Account, Groups, and Login. The main content area contains the following fields and options:

- Group:** incore_user
- Group Description:** incore_user
- NCSA Username (Requested):** A text input field with a red asterisk and a small icon to its right.
- First Name:** A text input field with a red asterisk.
- Last Name:** A text input field with a red asterisk.
- Organization:** A text input field.
- Primary Email:** A text input field with a red asterisk.
- Confirm Primary Email:** A text input field with a red asterisk.
- Agreement:** A checkbox with a red asterisk and the text 'I agree to the NCSA Terms of Service'.
- Submit Button:** A green button with a gear icon and the text 'Create and Join'.

At the bottom of the form, there is a footer with the following text:

All rights reserved. ©2018 Board of Trustees of the University of Illinois.
If you are having issues with Identity Services, please send an email to help+its@ncsa.illinois.edu

The NCSA logo is displayed at the bottom left of the page.

NOTE: Use your institutional email if possible.

- The username/password is used for accessing IN-CORE services. You can test your registration credentials by accessing the **IN-CORE page** at: <https://incore2.ncsa.illinois.edu/>.
- This is also used for accessing the **documentation** and downloading pyIncore package (**pyincore_<version>.tar.gz**) and Jupyter Notebook test file (**buildingdamage.ipynb**) at <https://incore2.ncsa.illinois.edu/>.

2 Installing pyIncore

2.1 Prerequisites

Please read through the instructions at least once completely before actually following them to avoid any installation problems!

IN-CORE account

- A user must have an IN-CORE account. If you don't have an account, see IN-CORE account section above.

Python 3.5+ (<https://www.python.org>)

- If you are on Windows, go to Windows 64 bit section.
- It is common to have more than one Python version installed on your computer. Make sure you are running the correct version of Python (you can check by running `python -version`) with corresponding path added to the `PATH` system variable. The following links will help you navigate through various installations guides
 - <https://realpython.com/installing-python/>
 - <https://docs.python-guide.org/#the-hitchhiker-s-guide-to-python>
 - OS specific downloads: <https://www.python.org/downloads/>

Virtual environment

We recommend that users get familiar with virtual environments (<https://www.pythonforbeginners.com/basics/how-to-use-python-virtualenv/>) or environment manager (<https://www.anaconda.com/distribution/>);

- These are tools that help keep dependencies separate for different projects. If you decide, however, to use a virtual environment or manager you must do it now, in this prerequisite step.
- A module named `virtualenv` is available by running `pip3 install virtualenv` (`pip3` command is `pip` for Python3, you could also run `pip3 install --upgrade pip` first),
OR
- An environment manager called Anaconda by downloading OS specific installer (<https://docs.anaconda.com/anaconda/install/>) Note that a full Anaconda distribution will include Python (and a collection of over 1,500+ open source packages), so installing Python first isn't needed if you use Anaconda. With Anaconda you already have installed Jupyter notebook. The `conda` is the preferred interface for managing installations and virtual environments with the Anaconda Python distribution.
- Note that `pip3` and `pip` would make a difference only when you are not using any environment managers. In `virtualenv` or `conda` environment which has `python==3.x`, `pip` would be equivalent to `pip3`.

Jupyter notebook (<https://jupyter.org/>)

We recommend using Jupyter Notebook for ease of running **pyIncCore** projects. It is an open-source application that allows you to create projects (documents) that contain live Python code, visualizations and documentation.

- Installing Jupyter (<https://jupyter.org/install.html>) can be done again with `pip` (on Miniconda; in your virtual environment) or `pip3` (no virtual environment) as indicated below:

```
pip install jupyter
```

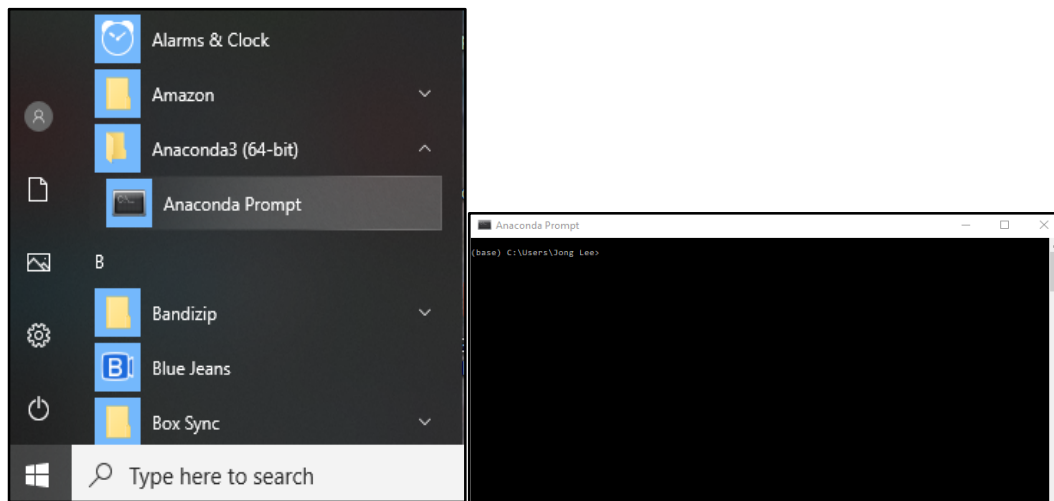
pyIncCore uses the Geospatial Data Abstraction Library (GDAL) (<https://www.gdal.org>), which has to be installed separately. Note that GDAL installation is global on Windows and Linux, even if you use virtual environments (see next step).

2.2 Windows 64-bit

GDAL (<https://www.gdal.org>) - Geospatial Data Abstraction Library

We provide installation instructions for Anaconda environment manager using Miniconda (<https://docs.conda.io/en/latest/miniconda.html>). Python 3.x and GDAL library will be installed with Miniconda. The following instructions were tested for Windows 64-bit (The 32-bit has not been tested yet):

1. Download the latest Miniconda3 installer for Windows from the Miniconda web page. (<https://docs.conda.io/en/latest/miniconda.html>)
2. Run the installer setup locally (select the “Just Me” choice) to avoid the need for administrator privileges.
3. Leave the default folder path (C:\Users\<user>\..\miniconda3).
4. Do not add Anaconda to the PATH. Do, however, register Anaconda as the default Python environment.
5. Open up an Anaconda prompt from the Windows Start menu. The `base` environment is being activated and the prompt changes to: `(base) C:\Users\<user>`



6. Create the python environment (pyincore for example) and activate it:

```
conda create -n pyincore python=3
conda activate pyincore
```

8. Install dependency packages in the following order:

```
conda install rasterio
conda install fiona
conda install rtree
conda install pyomo
conda install ipopt
```

pyIncCore package

These steps will guide you on how to install pyIncCore.

1. Download pyIncCore (pyincore-<version>.tar.gz) at <https://incore2.ncsa.illinois.edu/> to a directory on your computer.
2. To install pyIncCore, navigate to the directory you used on step 1 and:
From the Anaconda prompt run:
 - If you are using a virtual environment, you will need to activate it if it has not been done yet.
 - Run the following command:

```
pip install --user pyincore-<version>.tar.gz
```

pyIncCore credentials

The installation installs pyIncCore and creates an `.incore` folder in your HOME directory to store cached files. A message *pyIncCore credentials file has been created at <HOME directory>/.incore/.incorepw* appears in the prompt. The typical location of a HOME directory is `C:\Users\<username>` on Windows OS.

1. Locate a file called `.incorepw` in the `.incore` folder in your HOME directory.
2. Write your LDAP credentials in it; the first line contains your username and the second password. This information is used for communicating with IN-CORE web service.

2.3 Mac OS

GDAL (<https://www.gdal.org>) - Geospatial Data Abstraction Library

Use Homebrew (<http://mxcl.github.com/homebrew/>), a MacOS package manager. If you don't have Homebrew, please install it. Additional information about installing GDAL can be found at https://medium.com/@vascofernandes_13322/how-to-install-gdal-on-macos-6a76fb5e24a4

1. Install **gdal**:

```
brew install gdal
```
2. Install **spatialindex** library

```
brew install spatialindex
```
3. Also, update your pip Python package manager

```
pip3 install --upgrade pip
```

pyIncore package

These steps will guide you on how to install pyIncore.

1. Download pyIncore (`pyincore_<version>.tar.gz`) at <https://incore2.ncsa.illinois.edu/> to a directory on your computer.
2. To install pyIncore, navigate to the directory you used on step 1 and:
 - If you are using a virtual environment, you will need to activate it if it has not been done yet (here for Ananconda/Miniconda):

```
conda create -n pyincore python=3
```

```
conda activate pyincore
```
 - Run the following command:

```
pip install --user pyincore_<version>.tar.gz
```
 - We use the `matplotlib` library to create graphs. There is a Mac specific installation issue addressed at [StackOverflow](https://stackoverflow.com/questions/4130355/python-matplotlib-framework-under-macosx) <https://stackoverflow.com/questions/4130355/python-matplotlib-framework-under-macosx> and <https://stackoverflow.com/questions/21784641/installation-issue-with-matplotlib-python>. In a nutshell, insert line

```
backend: Agg
```

into the `~/.matplotlib/matplotlibrc` file. You must create the file (`matplotlibrc`) if it does not exist.

pyIncore credentials

The installation installs pyIncore and creates an `.incore` folder in your HOME directory to store cached files. A message *pyIncore credentials file has been created at <HOME directory>/.incore/.incorepw* appears in the prompt. The typical location of a HOME directory is `/Users/<username>` on Mac OS.

Note: The folders and files starting with "." (dot prefix) are hidden in Operating systems with Unix roots. There are few ways (<https://nektony.com/how-to/show-hidden-files-on-mac> and <https://macpaw.com/how-to/show-hidden-files-on-mac>) to view hidden files on your Mac.

1. Locate a file called `.incorepw` in the `.incore` folder in your HOME directory.
2. Write your LDAP credentials in it; the first line contains your username and the second password. This information is used for communicating with IN-CORE web service.

2.4 Linux

GDAL (<https://www.gdal.org>) - Geospatial Data Abstraction Library

Additional information about installing GDAL can be found at <https://github.com/domlysz/BlenderGIS/wiki/How-to-install-GDAL>).

1. Install **gdal-bin**:

```
sudo apt-get install gdal-bin
```
2. Install **libspatialindex-dev** library

```
apt-get install libspatialindex-dev
```
3. Also, update your pip Python package manager

```
pip3 install --upgrade pip
```

pyIncore package

These steps will guide you on how to install pyIncore.

1. Download pyIncore (`pyincore_<version>.tar.gz`) at <https://incore2.ncsa.illinois.edu/> to a directory on your computer.
2. To install pyIncore, navigate to the directory you used on step 1 and:
 - If you are using a virtual environment, you will need to activate it if it has not been done yet (here for Ananconda/Miniconda):

```
conda create -n pyincore python=3
```

```
conda activate pyincore
```

- Run the following command:

```
pip install --user pyincore_<version>.tar.gz
```
- If you see the *OSError: Could not find libspatialindex_c library file* error, make sure that you installed **libspatialindex-dev** (see GDAL section above).

pyIncore credentials

The installation installs pyIncore and creates an `.incore` folder in your HOME directory to store cached files. A message *pyIncore credentials file has been created at <HOME directory>/.incore/.incorepw* appears in the prompt. The typical location of a HOME directory is `/home/<username>` on Linux based machines.

Note: The folders and files starting with "." (dot prefix) are hidden in Operating systems with Unix roots.

1. Locate a file called `.incorepw` in the `.incore` folder in your HOME directory.
2. Write your LDAP credentials in it; the first line contains your username and the second password. This information is used for communicating with IN-CORE web service.

3 Testing pyIncore Installation

- For these instructions we assume that users develop their python script by using pyIncore in their own **project folder** (create folder if you don't have one)
- Download the Jupyter Notebook file for Building damage analysis (<https://incore2.ncsa.illinois.edu/doc/examples/buildingdamage.ipynb>) to your **project folder**. We will verify your installation of pyIncore by running this file.

3.1 Running a Building Damage Analysis Locally

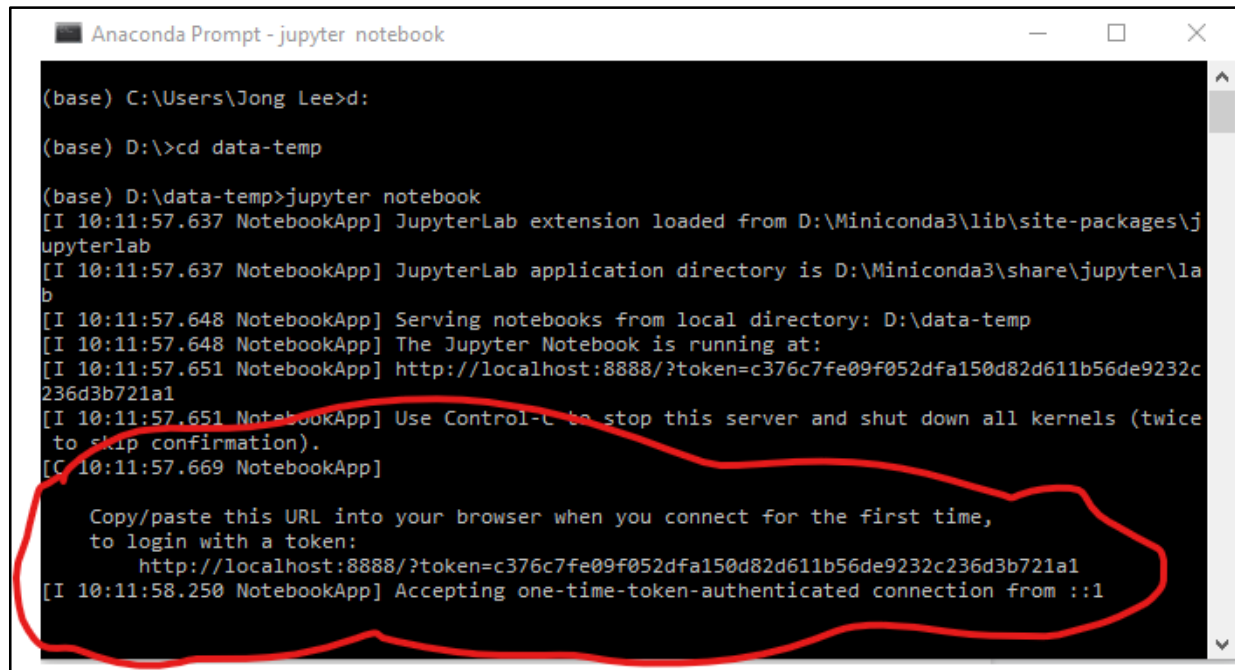
- Start a local **Jupyter Notebook** by running the following command in the terminal or command prompt from your **project folder** (change directories to the particular project folder at the command prompt):

```
jupyter notebook
```

or if **Jupyter Notebook** is not recognized in Anaconda

```
python -m notebook
```


A message *The Jupyter Notebook is running* appears in the terminal/prompt and you should see the notebook dashboard open in your browser. Note that you might be asked to copy/paste a URL into your browser when you connect for the first time as shown below:



```
Anaconda Prompt - jupyter notebook

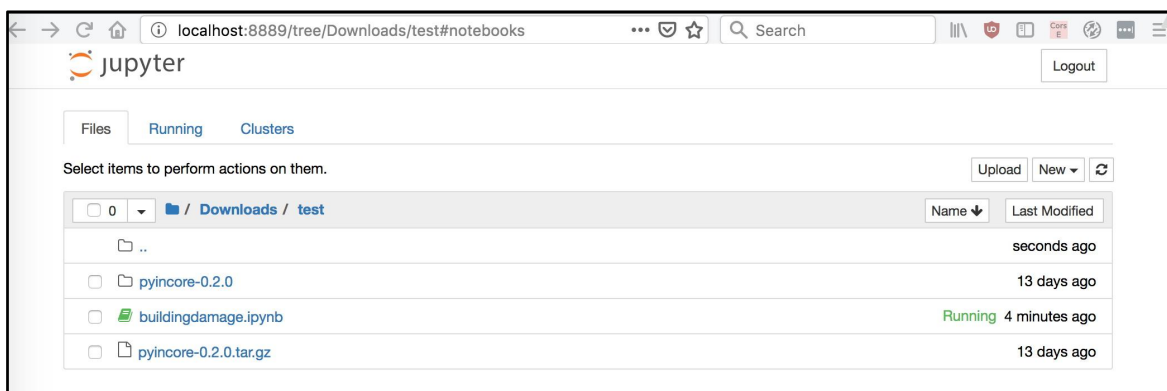
(base) C:\Users\Jong Lee>d:

(base) D:\>cd data-temp

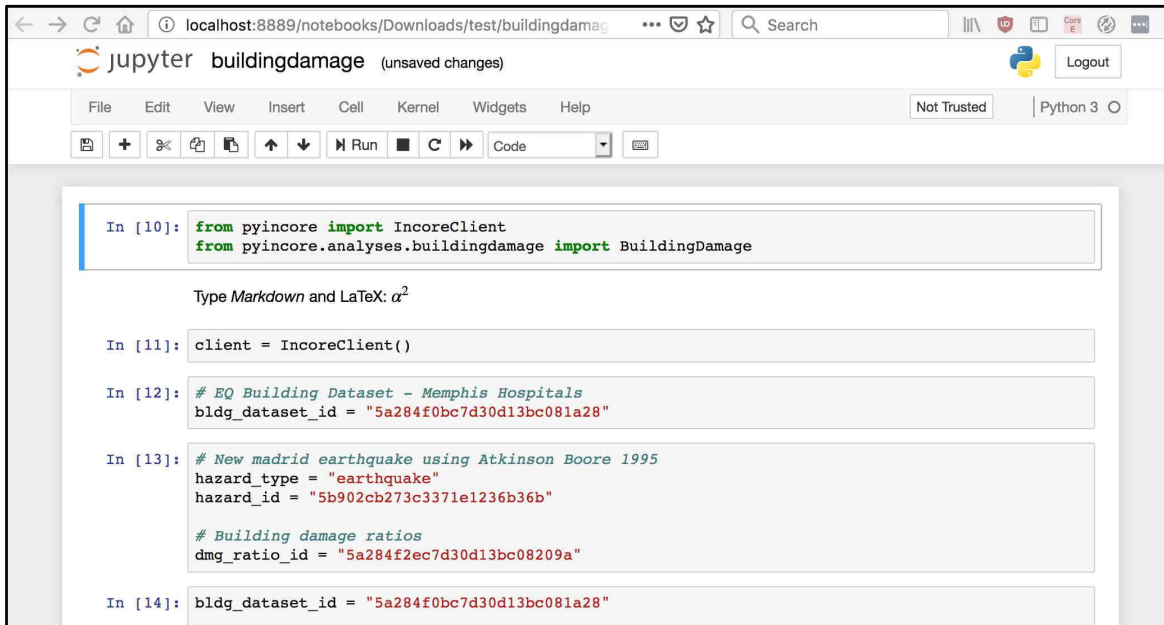
(base) D:\data-temp>jupyter notebook
[I 10:11:57.637 NotebookApp] JupyterLab extension loaded from D:\Miniconda3\lib\site-packages\jupyterlab
[I 10:11:57.637 NotebookApp] JupyterLab application directory is D:\Miniconda3\share\jupyter\lab
[I 10:11:57.648 NotebookApp] Serving notebooks from local directory: D:\data-temp
[I 10:11:57.648 NotebookApp] The Jupyter Notebook is running at:
[I 10:11:57.651 NotebookApp] http://localhost:8888/?token=c376c7fe09f052dfa150d82d611b56de9232c236d3b721a1
[I 10:11:57.651 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 10:11:57.669 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
    http://localhost:8888/?token=c376c7fe09f052dfa150d82d611b56de9232c236d3b721a1
[I 10:11:58.250 NotebookApp] Accepting one-time-token-authenticated connection from ::1
```

- Click on the `buildingdamage.ipynb` in the Jupyter Notebook browser.



Your web page should now show multiple cells of code like this:



Right now you are not actually running a notebook yet. Running a cell means that you will execute the cell's contents. To execute cells in order you can just select the first cell and click the **Run** button at the top.

Note that **Building damage** is a long running analysis and there is little indication that it's running except by either looking at the Jupyter Notebook file and seeing the [*] for the notebook cell where that block of code is being executed or by looking at the Task Manager in the Notebook dashboard to see there is a python process running. Alternatively, you can look at the Jupyter Notebook dashboard to see if the `csv` file with results has been created yet.

For details of running and manipulating `ipynb` files refer to Jupyter documentation (<https://jupyter.readthedocs.io/en/latest/running.html#running>).

4 Using IN-CORE Lab

IN-CORE Lab is a customized Jupyter Lab for running and editing Notebooks accessible at <https://incore-lab.ncsa.illinois.edu>.

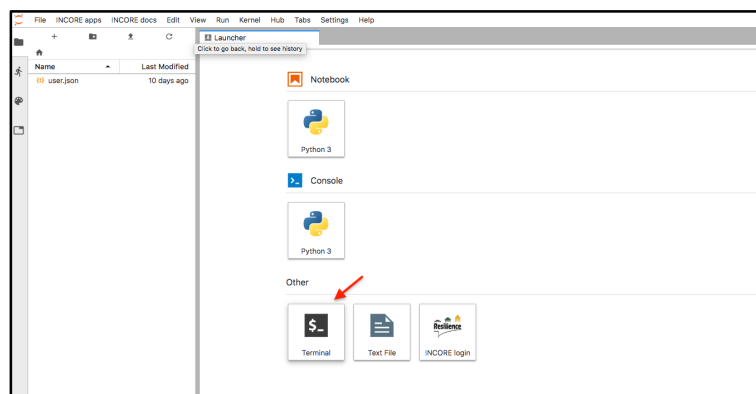
- Login to IN-CORE Lab with you IN-CORE account info (you created at the beginning of this session)

A sign-in form with an orange header bar containing the text "Sign in". Below the header, there are two input fields: "Username:" and "Password:". Each field has a small "x" icon on the right side. At the bottom of the form is an orange button labeled "Sign In".

4.1 Running Jupyter Notebook in IN-CORE Lab

In *Testing pyIncore Installation* section we described how to run Building damage Notebook locally. This section focuses on step-by-step instructions of running Notebooks on the IN-CORE Lab.

1. Create a credential file with IN-CORE username/password (same information you used to login to IN-CORE Lab) in order to use IN-CORE services. This is similar to the authentication step described in *Testing pyIncore Installation* section except the authentication file `.incorepw` is being created on the IN-CORE Lab server running Linux OS:
 - a. Open the terminal on IN-CORE lab Launcher page:



- b. In the terminal, make sure you are in your HOME directory. Type:
`pwd`

to see the current path and

```
cd ~
```

to get into your home directory (/home/<username>).

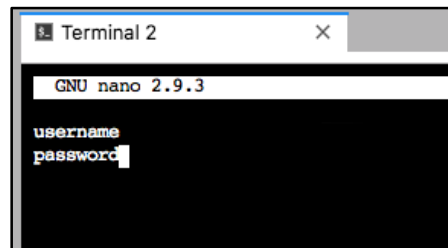
- c. Create a hidden (therefore dot prefix) folder:

```
mkdir .incore
```

- d. Create a hidden credential file in the folder you just created and type IN-CORE username and password using **nano** text editor:

```
cd .incore
```

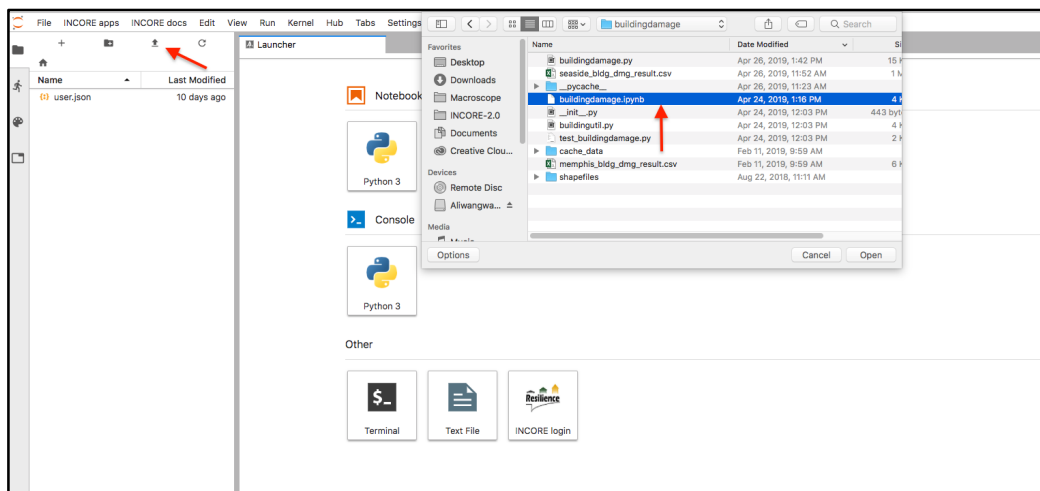
```
nano .incorepw
```



- e. Save the file with **Ctrl+O** and **Enter** commands

- f. Close the text editor and return to your shell with **Ctrl+X** command

2. Upload the Building Damage Notebook from your local machine to IN-CORE lab by clicking the **Upload** icon in the left panel and select **buildingdamage.ipynb**.



3. The building damage Notebook shows up in the left panel after a successful upload.
4. Double click to open it in the main area.

```

from pyincore import IncoreClient
from pyincore.analyses.buildingdamage import BuildingDamage

Type Markdown and LaTeX:  $\alpha^2$ 

[ ] client = IncoreClient()

[ ] # FO Building Dataset - Memphis Hospitals
bldg_dataset_id = "5a284f8c7d38d13bc803a28"

[ ] # New madrid earthquake using Atkinson Boore 1995
hazard_type = "earthquake"
hazard_id = "50902cb273c3371e1236b36b"

# Building damage ratios
dmg_ratio_id = "5a284f8c7d38d13bc803a28"

[ ] bldg_dataset_id = "5a284f8c7d38d13bc803a28"

[ ] # Earthquake mapping
mapping_id = "5a47b3583376a63d2987672c"

[ ] # Run Memphis earthquake building damage
bldg_dmg = BuildingDamage(client)
bldg_dmg.load_remote_input_dataset("buildings", bldg_dataset_id)
bldg_dmg.load_remote_input_dataset("dmg_ratio", dmg_ratio_id)

result_name = "memphis_bldg_dmg_result"
bldg_dmg.set_parameter("result_name", result_name)
bldg_dmg.set_parameter("mapping_id", mapping_id)
bldg_dmg.set_parameter("hazard_type", hazard_type)
bldg_dmg.set_parameter("hazard_id", hazard_id)
bldg_dmg.set_parameter("num_cpu", 1)

# Run Analysis
bldg_dmg.run_analysis()

[ ] # Seaside example Tsunami
hazard_type = "tsunami"
hazard_id = "5b3c2b2e7b88533c7e6184c"

[ ] # Seaside building dataset
bldg_dataset_id = "5bcf2fcd7242fe847ce79dad"

```

- Run it. Instructions on how to run building damage analysis, please refer to previous section *Running a Building Damage Analysis*.

5 How to Contact and Work with NCSA

- Contact an individual programmer developer by email and copy incore-dev@lists.illinois.edu if you work closely with NCSA on a code conversion and/or improvement of your hazard analysis.
- Contact the incore-dev@lists.illinois.edu email list if you do not work directly with NCSA.
- Response time during the week will be in approximately 24 hours or less. Weekend emails will be responded to on the next business day, typically Monday.

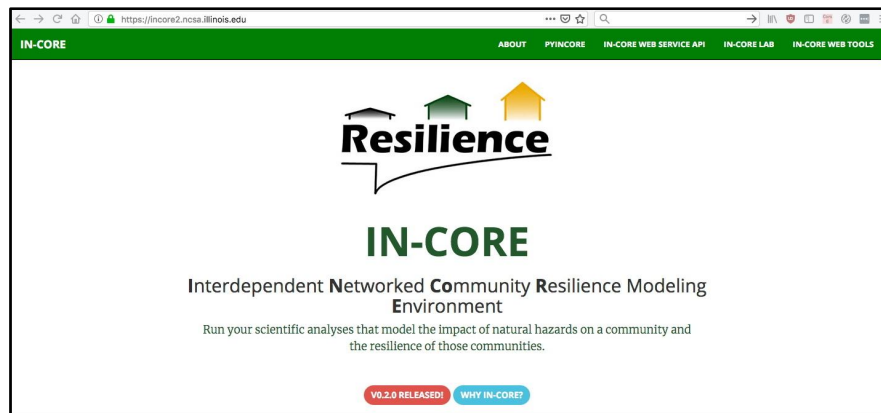
6 Information for pyIncore Developers

- IN-CORE programming guideline:
<https://opensource.ncsa.illinois.edu/confluence/display/INCORE1/IN-CORE+Programming+Guideline>
- Python programming tips:
<https://opensource.ncsa.illinois.edu/confluence/display/INCORE1/Programming+Tips+for+Python>

7 Additional Information

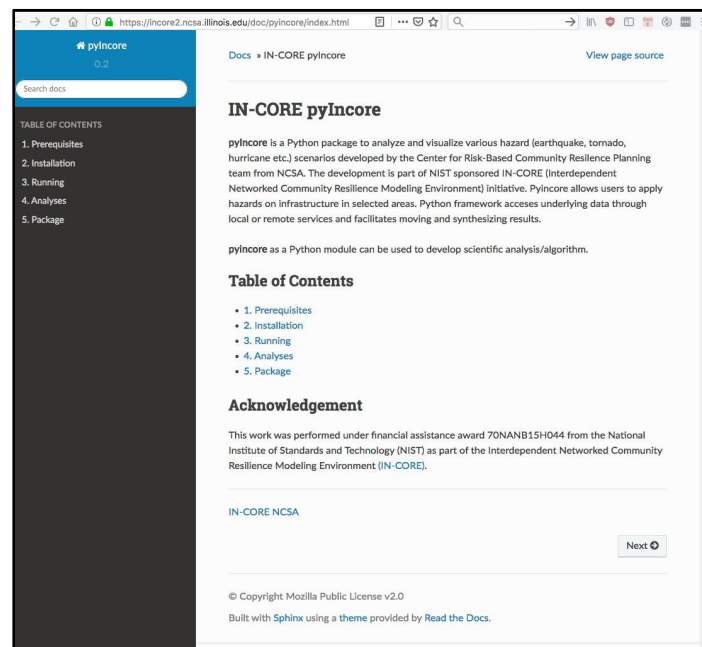
7.1 Technical Documentation

From the IN-CORE landing page at <https://incore2.ncsa.illinois.edu/> a user can access:

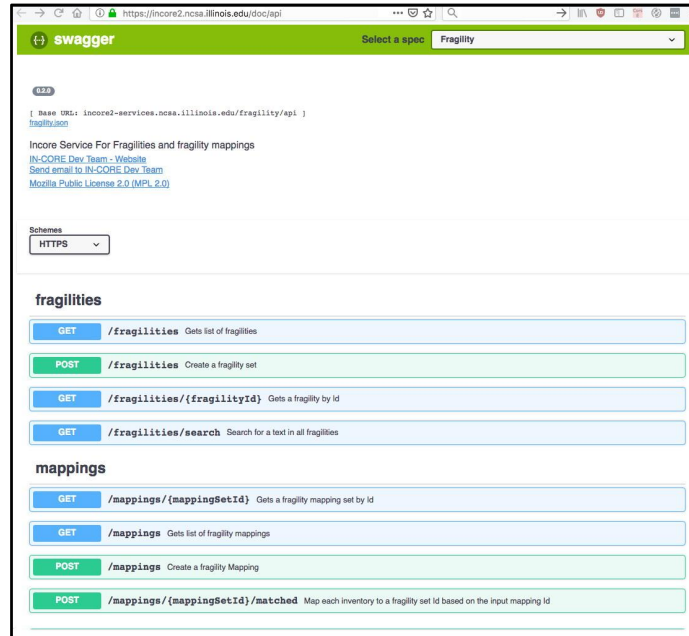


In this site, you can download pyIncore package and access to other services and documentations.

- Documentation of **pyIncore** is at <https://incore2.ncsa.illinois.edu/doc/pyincore/>



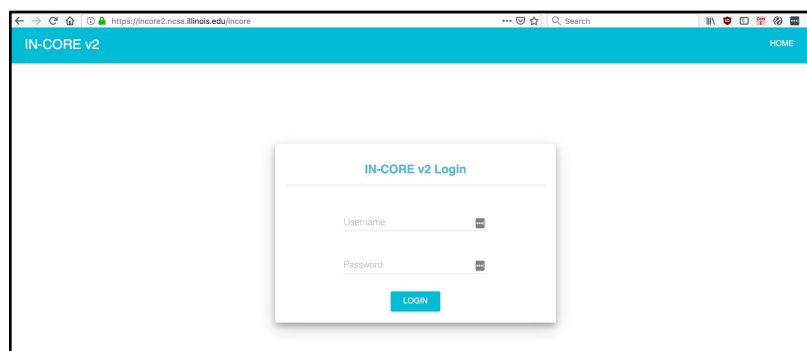
- Documentation of **IN-CORE Web Service** is at <https://incore2.ncsa.illinois.edu/doc/api/>



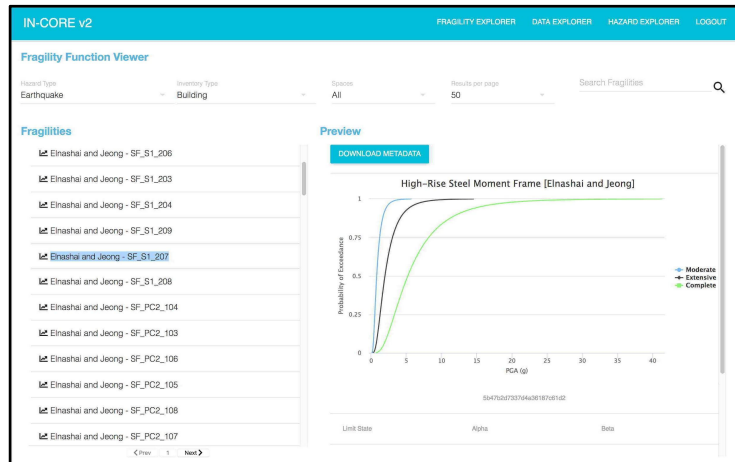
- IN-CORE Lab is at <https://incore-lab.ncsa.illinois.edu>
- IN-CORE Web Tools are for interacting with the service layer. They enable users to browse and search the **Datasets**, **Hazards** and **Fragilities**, view the metadata and visualizations, and download the datasets.

7.2 IN-CORE Web tools

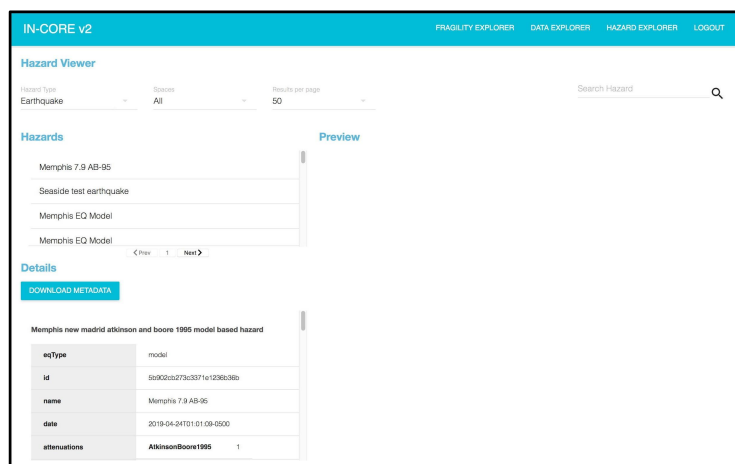
In-Core Web tools are a dashboard with viewers for various services. Currently these are **Fragility**, **Data** and **Hazard** services. A user must login with IN-CORE username and password in order to access the viewers:



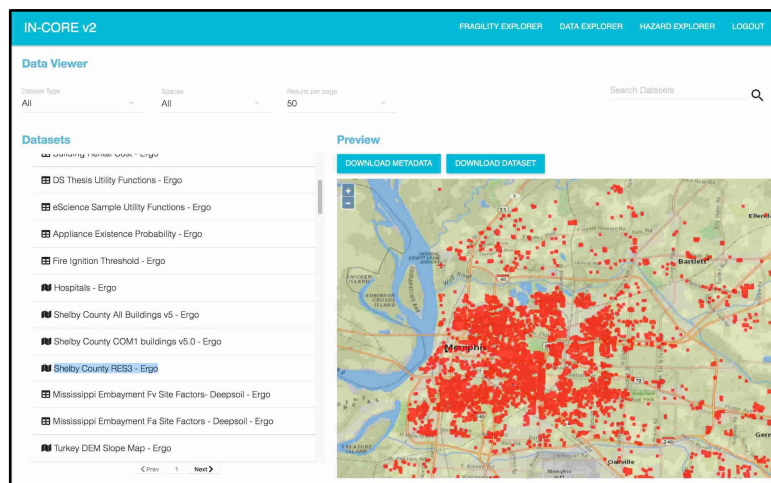
Fragility service. This is an example of a viewer showing a selection list (left) of Fragility curves. Hazards and types of structures are selected in the pull down menus. The data can be downloaded in `json` format.



Hazard service viewer.



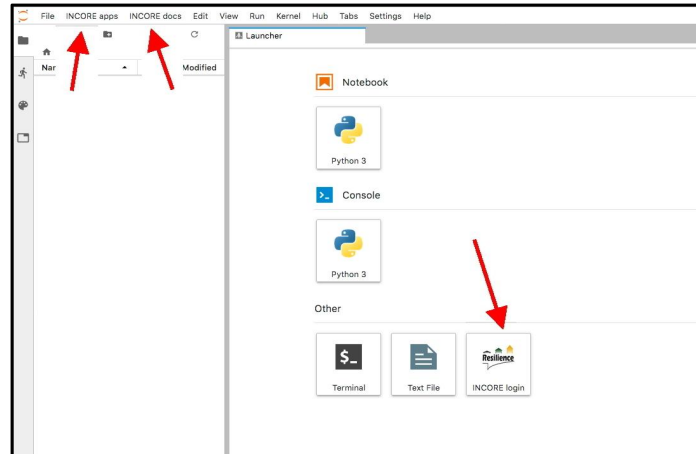
Data service viewer.



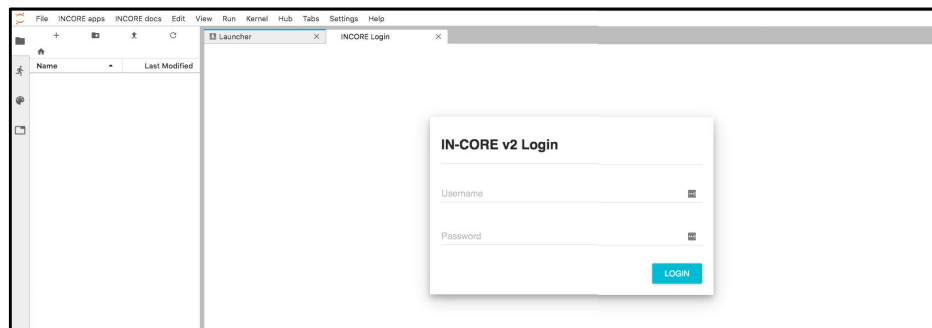
7.3 Additional Information about IN-CORE Lab

This section shows how to access IN-CORE Web Tools and documentations on IN-CORE Lab.

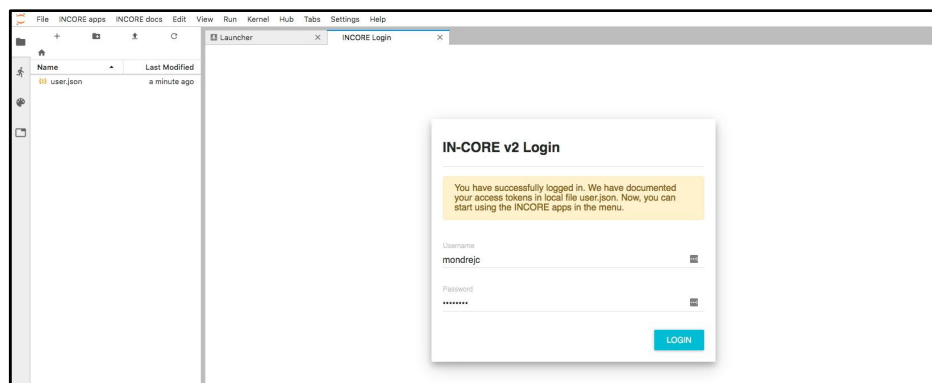
- Click on “INCORE Login” button as shown below.



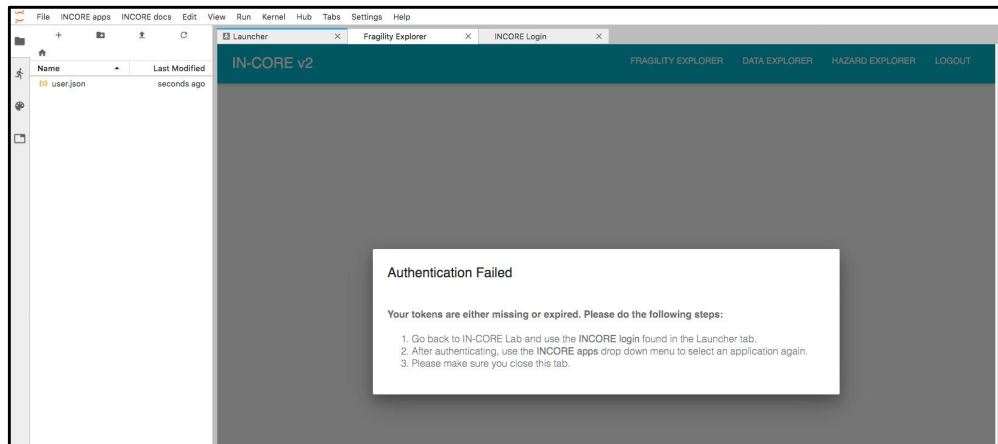
Same username and password for this part.



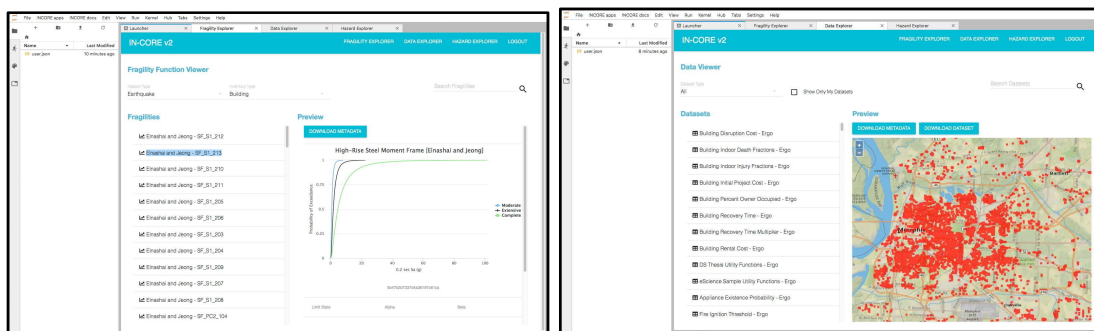
This login process generates a file named **user.json**. It appears in the File list manager on the left side. The file contains an authentication token required for development of new analyses using IN-CORE’s Application programming interface (API).



Fragility, Data and Hazard Explorers under INCORE apps menu become enabled after pressing LOGIN button AND reloading the current page in the browser. **NOTE:** A user must reload the whole Jupyter dashboard page (above) using the Reload button of the browser, not the Refresh File List (part of Jupyter's file navigation) otherwise a following Warning appears:



Viewer as part of INCORE Lab as shown below.



- The second IN-CORE menu (INCORE docs) allows user to see pyIncore documentation and API endpoints definitions for accessing Fragility, Data and Hazard server(s). Another IN-CORE login window opens up at the top of the browser's main window.

For ease of access - documentation is easily accessible from IN-CORE Lab.

