



# DataCite Metadata



*Excel → XML Script*

---

## – **Python script:** `dataCiteExcelToXML.py`

1. Takes all data from the Excel workbook and creates a single CSV

2. Finds all distinct JHED-Request# identifiers in the CSV

3. For each distinct identifier, the script loops through each row in the CSV

4. In rows where it finds that identifier, the associated element/data is mapped into XML

5. After it loops through all rows, creates XML document for that identifier

6. Repeats for each distinct identifier

7. Done!

# Let's take a look

- Lines 32-44 ; Creates the CSV
- Lines 48-54 ; Finds all unique JHED-Request numbers
- Lines 60-533 ; Maps data to XML elements & attributes based on identifier (repeatable loop)
- Lines 533-542; Creates XML document (repeatable loop)

# What does “dataCiteExcelToXML.py” do?

## Capabilities

1	Transfers all data exactly as entered from Excel workbook to new XML document(s)	✓
2	Matches Excel fields to corresponding XML element/attribute in the DataCite Schema 4.2	✓
3	Formats schema & namespace declarations in resource element	✓
4	Removes extra spaces at the end of data strings	✓

## Limitations

1	Does not confirm data in Excel adheres to data types or schema standards	✗
2	Does not automatically validate the XML files	✗

# Tips: Filling-out Excel workbook

- **Each row equals one second-level element**

Examples: Name, Contributor, Subject, Date, Funder, etc. **The sole exception to this rule is GeoLocation (see slide 6-8)**

- **Each row with data, no matter what, must have an JHED-Request#**

If there is no request number, the information in that row will not be added to any XML document

# GeoLoc mapping

- All information from *same row* OR with the *same geoLocationPolygon identifier* is considered part of *one* geoLocation
- For PolygonPoint #2, 3, 4, etc. after the initial point, only fill out Polygon information in those row
- If you want multiple geoLocations, use multiple rows

Example one: 1 geoLocation



A	B	C	D	F	G	H	I	J	K	L	M	N
JHED - Request#	geoLocationPoint	pointLongitude	pointLatitude	westBound	eastBound	southBound	northBound	geoLocationPlace	geoLocationP	polygonPoint	polyPoint	polyPoint
agnanad1-1	31.233, -67.302	-67.302	31.233	-71.032	-68.211	41.09	42.893	Atlantic Ocean	polygon1	41.991, -71.032	-71.032	41.991
agnanad1-1									polygon1	42.893, -69.622	-69.622	42.893
agnanad1-1									polygon1	41.991, -68.211	-68.211	41.991
agnanad1-1									polygon1	41.090, -69.622	-69.622	41.09
agnanad1-1									polygon1	41.991, -71.032	-71.032	41.991

Example two: 3 geoLocations



8	agnanad1-1								polygon2	0.210307, 51.485	0.210307	51.4859
9	agnanad1-1								polygon2	0.211056, 51.485	0.211056	51.4859
10	agnanad1-1								polygon2	0.211517, 51.485	0.211517	51.4859
11	agnanad1-1								polygon2	0.212561, 51.485	0.212561	51.4858
12	agnanad1-1								polygon2	0.210307, 51.485	0.210307	51.4859
13	agnanad1-1	60.62458, -1.31100	-1.31100	60.62458				Scotland				
14	agnanad1-1							Democratic Republic of the Congo				

# GeoLocation from Example 1



```
<geoLocation>
  <geoLocationPlace>Atlantic Ocean</geoLocationPlace>
  - <geoLocationPoint>
    <pointLongitude>-67.302</pointLongitude>
    <pointLatitude>31.233</pointLatitude>
  </geoLocationPoint>
  - <geoLocationBox>
    <westBoundLongitude>-71.032</westBoundLongitude>
    <eastBoundLongitude>-68.211</eastBoundLongitude>
    <southBoundLatitude>41.09</southBoundLatitude>
    <northBoundLatitude>42.893</northBoundLatitude>
  </geoLocationBox>
  - <geoLocationPolygon>
    - <polygonPoint>
      <pointLongitude>-71.032</pointLongitude>
      <pointLatitude>41.991</pointLatitude>
    </polygonPoint>
    - <polygonPoint>
      <pointLongitude>-69.622</pointLongitude>
      <pointLatitude>42.893</pointLatitude>
    </polygonPoint>
    - <polygonPoint>
      <pointLongitude>-68.211</pointLongitude>
      <pointLatitude>41.991</pointLatitude>
    </polygonPoint>
    - <polygonPoint>
      <pointLongitude>-69.622</pointLongitude>
      <pointLatitude>41.09</pointLatitude>
    </polygonPoint>
    - <polygonPoint>
      <pointLongitude>-71.032</pointLongitude>
      <pointLatitude>41.991</pointLatitude>
    </polygonPoint>
  </geoLocationPolygon>
</geoLocation>
```

# GitHub

- This script and associated information currently lives in [my GitHub](#)
  - To run, download the folder to your computer
  - We can also make a shared Github to keep this info if desired

The screenshot shows a GitHub repository page for 'mjanowiecki / datacite'. The repository is private and has 15 commits, 1 branch, and 0 releases. The repository description is 'No description or website provided.' The repository contains several files: .gitattributes, .gitignore, README.md, convertExcelSheetsToCSV.py, dataCiteExcelToXML.py, and metadataDummy.xlsx. The README.md file is selected, showing the title 'DataCite Repository' and the description 'Python scripts to help researchers submit valid XML documents to DataCite to create DOIs and their metadata.' The repository also has a file named 'dataCiteExcelToXML.py' with the description 'This script creates well-formed XML documents for importation into DataCite from an Excel workbook.'

mjanowiecki / datacite Private

Unwatch 1 Star 0 Fork 0

<> Code Issues 0 Pull requests 0 Projects 0 Insights Settings

No description or website provided. Edit

datacite-xml lxml excel-to-xml Manage topics

15 commits 1 branch 0 releases

Branch: master New pull request Create new file Upload files Find File Clone or download

mjanowiecki Update README.md Latest commit c326ad7 3 hours ago

.gitattributes	Initial commit	a day ago
.gitignore	Create .gitignore	a day ago
README.md	Update README.md	3 hours ago
convertExcelSheetsToCSV.py	Initial commit	a day ago
dataCiteExcelToXML.py	Update readability	3 hours ago
metadataDummy.xlsx	Update readability	3 hours ago

README.md

## DataCite Repository

Python scripts to help researchers submit valid XML documents to DataCite to create DOIs and their metadata.

### dataCiteExcelToXML.py

This script creates well-formed XML documents for importation into DataCite from an Excel workbook.



# Overview: Running Python scripts

**Complete setup on your work computer (I can help/provide further instructions if we pick this)**

→ **Open up script in an text editor (i.e. Atom, Notepad++)**

This is optional, but can help us see the code if needed.

→ **Move the filled-out Excel file into the folder where [dataCiteExcelToXML.py](#) lives**

This makes sure the script can successfully find the workbook

→ **Open up your terminal**

This is how we will run the code

→ **Run [dataCiteExcelToXML.py](#) in your terminal**

This converts our spreadsheet into a single CSV document and then into one or multiple XML documents.

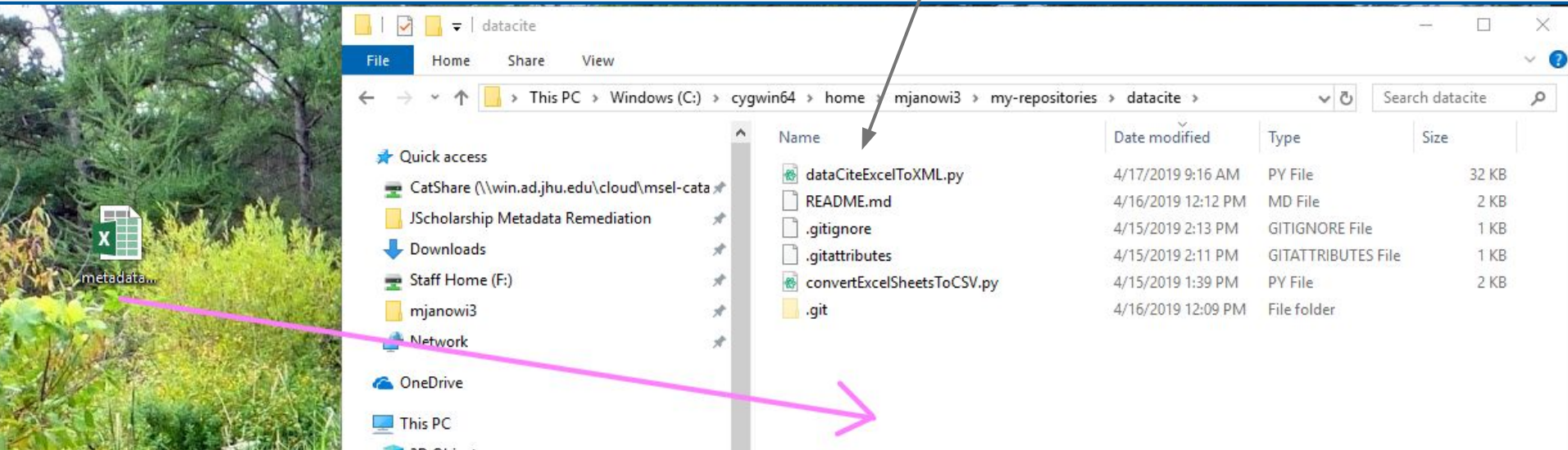
→ **Conversion done! You are ready to validate and upload to DataCite**

# Step 1:

Put everything  
in the same  
place

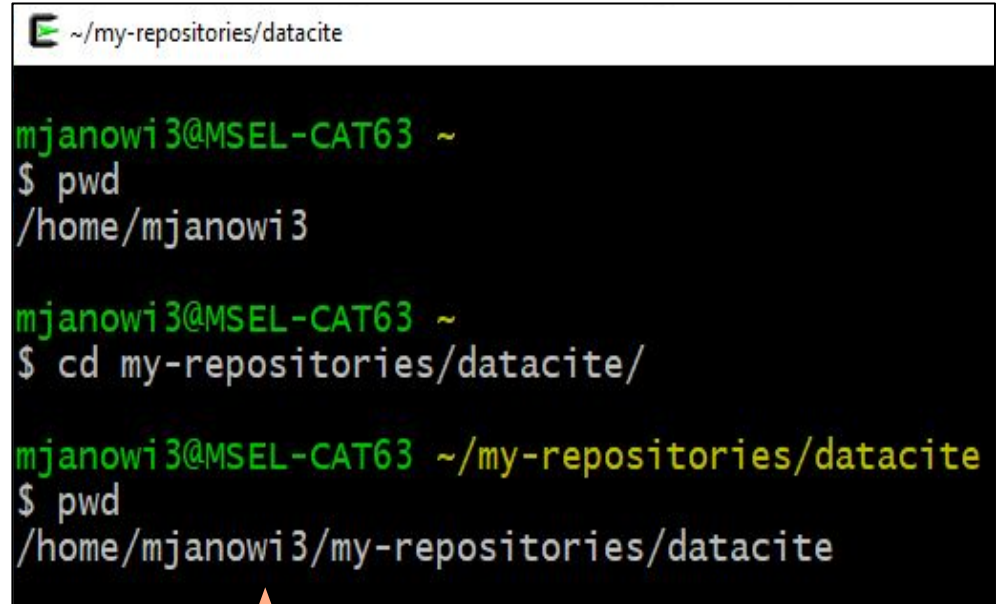
Where does your Python script live?

Before running the script, be sure to place the filled-out Excel workbook in the folder where dataCiteExcelToXML.py lives!



## Step 2: Navigate to the folder on your terminal where the script is located

- Open your terminal
- `pwd`
  - Means “print working directory”
  - Gives your current folder location
- `cd`
  - Means “change directory”
  - It allows you to move to another folder within your current directory
  - Asking to move two folders down in the “mjanowi3” folder:  
mjanowi3-->my-repositories-->datacite
  - Result? When we enter `pwd` again, we are in the right folder!



```
mjanowi3@MSEL-CAT63 ~  
$ pwd  
/home/mjanowi3  
  
mjanowi3@MSEL-CAT63 ~  
$ cd my-repositories/datacite/  
  
mjanowi3@MSEL-CAT63 ~/my-repositories/datacite  
$ pwd  
/home/mjanowi3/my-repositories/datacite
```

The terminal window shows the user navigating from their home directory to the `my-repositories/datacite` directory. The title bar indicates the current path is `~/my-repositories/datacite`. An orange arrow points from the text "we are in the right folder!" to the final `pwd` output.

## — Step 3: Run the script!

After navigating to the folder with the script, type:

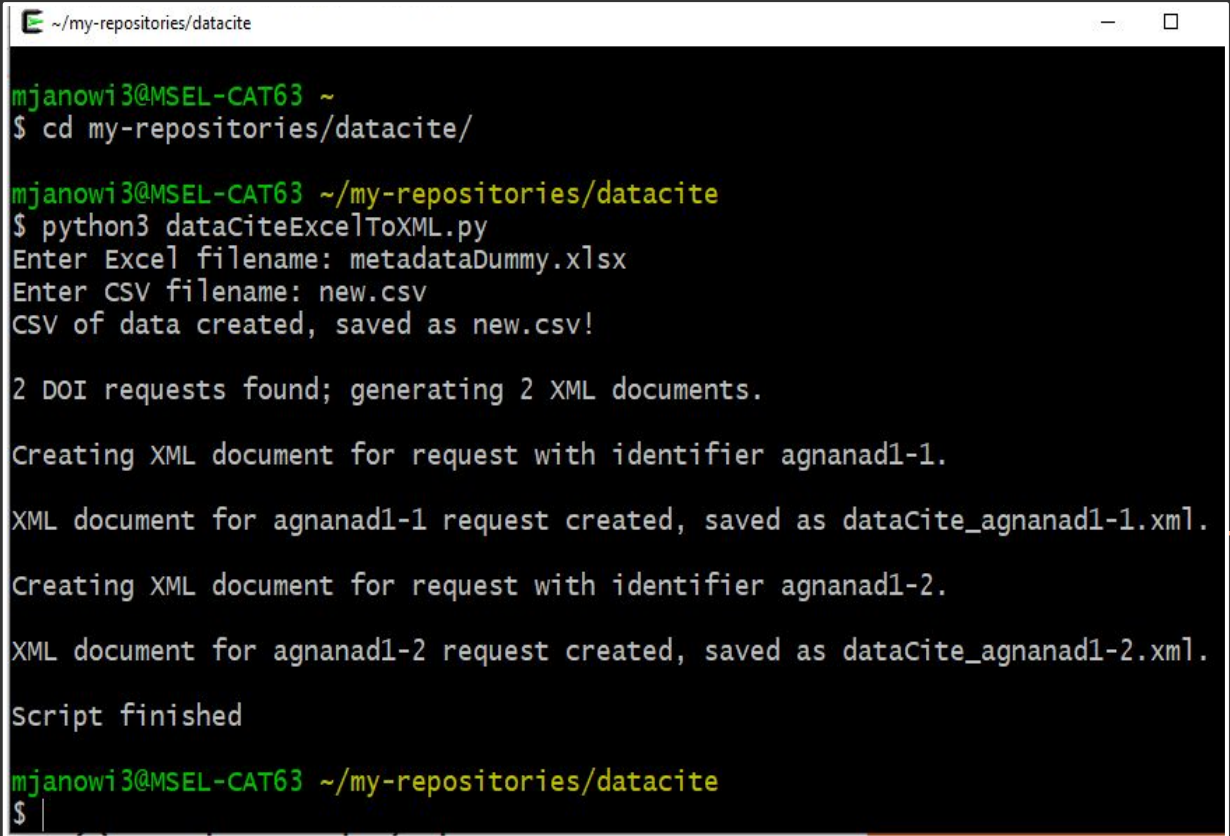
`python3 dataCiteExcelToXML.py`

and press enter!

Answer the prompts

Yay! The script is running.

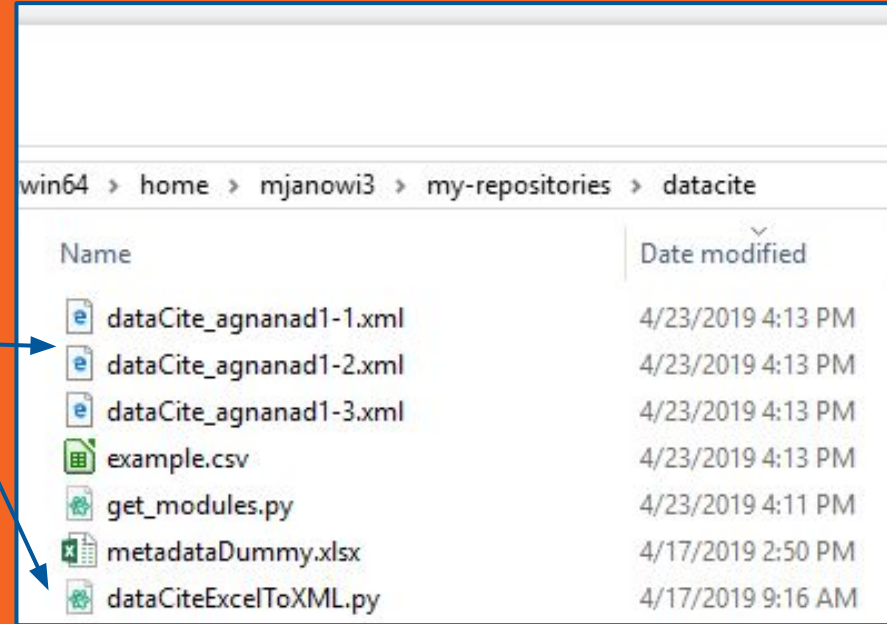
When you see a new \$, your script is finished



```
mjanowi3@MSEL-CAT63 ~  
$ cd my-repositories/datacite/  
  
mjanowi3@MSEL-CAT63 ~/my-repositories/datacite  
$ python3 dataCiteExcelToXML.py  
Enter Excel filename: metadataDummy.xlsx  
Enter CSV filename: new.csv  
CSV of data created, saved as new.csv!  
  
2 DOI requests found; generating 2 XML documents.  
  
Creating XML document for request with identifier agnanad1-1.  
  
XML document for agnanad1-1 request created, saved as dataCite_agnanad1-1.xml.  
  
Creating XML document for request with identifier agnanad1-2.  
  
XML document for agnanad1-2 request created, saved as dataCite_agnanad1-2.xml.  
  
Script finished  
  
mjanowi3@MSEL-CAT63 ~/my-repositories/datacite  
$
```

# Viewing your newly created documents:

- The XML documents generated will be put in the folder where you keep `dataCiteExcelToXML.py`
- To view the XML, open with web browser or with Atom
- Your CSV of all of metadata has also been generated



---

# Troubleshooting and errors

## *Error messages in the terminal*

### → “[Errno 2] No such file or directory”

This is mostly happens when you make a typo when entering file names or you are in the wrong folder

### → “[Errno 16] Device or resource busy:”

You need to close whatever file it indicates here.

## *Problems with XML documents*

### → I have more XML documents than identifiers

This is mostly likely caused by a typo or a blank cell in the JHU-Request# in the Excel file

### → Some elements have unexpected values in them

Excel sometimes treats dates and number values as math problems or removes leading values. Trying formatting the problem cell as 'Text.'

### → Some of my special characters are now random strings

Try opening the XML file in a browser like Chrome, if you are able to read the characters there, it should work when you upload on DataCite.

### → Some of the values from the Excel spreadsheet are missing in the XML

For attributes and elements that are required as a combo (like `nameIdentifier` & `nameIdentifierScheme`), the script requires both of them in order to add them to the XML. Double check the Excel that all the data is there.

---

# Setup: A Reference Guide

## Windows, Part 1:

Install [Cygwin \(64 bit version\)](#) & [Atom](#)

### *Installing Cygwin*

- Download and run the installer
- Keep pre-selected defaults and hit next on all windows until you reach the “Select Packages” window
- Select the python36 language to add, along with the following packages, all prefixed by “python36-”. Add any associated “debuginfo” modules. Finish the installation.

- appdirs
- asn1crypto
- cffi
- chardet
- cryptography
- Cython
- idna
- lxml
- numpy
- olefile
- packaging
- pathlib2

- pip
- ply
- pyasn1
- pycparser
- pyparsing
- pytz
- requests
- setuptools
- simplejson
- six
- urllib3
- wheel

# Setup: A Reference Guide Cont.

## Windows, Part 2:

### *Installing remaining packages through the terminal*

- Open Cygwin terminal
- Type “pip3 install dateutil” & hit enter
- Type “pip3 install xlrd” & hit enter
- Type “pip3 install pandas” & hit enter
- Set-up complete. After this, you should be ready to run the script.