

SpredNonDicomUpload Documentation

Introduction

This is some brief documentation for the work I did for the Mouse Imaging Centre (MICe) and the Ontario Brain Institute (OBI). The OBI is working on a big data solution for neuroscience data, called [Brain-CODE](#), and it was my job to upload approximately 50 GB of mouse brain imaging data from MICe into the imaging side of Brain-CODE. The imaging side of Brain-CODE uses the Stroke Patient Recovery Research Database (SPReD).

Right now the upload process uses an R script to generate the subject metadata, then uses a Python script to map the subject metadata to the XNAT XML schema, to extract other metadata, and to structure HTTP PUT web service calls to the XNAT REST API.

To learn how to use these scripts, look at the Workflow below and make sure you have everything installed listed in the Dependencies section. Development Notes / Rationales contains information for developers who might be interested in extending this work (maybe OOP-ifying it).

Workflow

1. Specify folders that you want to upload along with their corresponding gf files and strain label (label used as a prefix for all the data files – e.g. NLGN) in the GenerateSubjectMetadata.R script. Run the script.
2. Move the .csv file that is produced to the same directory where you will run the Python script.
3. Modify globals in the Python script to specify the project name, site code, SPReD url, and folders that you want to upload to SPReD.
4. Run the Python script, supplying your username and password, and it will upload subjects, sessions, scans, and any associated files to the project and site at the specified SPReD url. By default, the script is interactive to help ensure that the proper files are uploaded.

Dependencies

- mincinfo from minc-tools (<https://github.com/BIC-MNI/minc-tools>)
- python packages (datetime, numpy, os, pandas, requests, subprocess, sys)

Development Notes / Rationales

Incorrect Field Mappings This section describes some MINC fields that were difficult to map to permissible XNAT XML elements and attributes.

- the difference between `study:start_time` and `vnmr:time_submitted` in the MINC headers is unknown
 - it is unclear whether these fields would map to SPReD session and scan, respectively
 - for MICe’s purposes, there will only ever be 1 scan per session, so it doesn’t matter for the current implementation
- fov (x and y) in XNAT is of type integer, yet these variables are of type float in the MINC header
 - to convert to mm, simply multiplied by 10

- fov and matrix fields in XNAT are two dimensional (only x and y), however, all images at MICE are 3 dimensional without slices
 - what is the best way of handling this behaviour? possibly supply z dimension in the **frames** XNAT XML element
- duration goes up to a maximum of around 16 hours
 - for some reason XNAT is not accepting well-formed xs:duration values
- it's not clear what fields to use for mouse background, strain, and genotype

Rationales

- upload process is now interactive
 - when you're uploading such a large amount of data someone should probably oversee the process
 - the script is safe to run without interactivity because it will automatically shut off if it fails during one step of the process, and will notify the user what subject and action (create subject, session, scan, resource, file) that it failed on
 - a bit torn on whether interactive mode should be optional or a command line argument
- decided not to refactor methods so that instead of passing spread id and MINC filename around, you pass an object or namedtuple, because it would just be extra work
- extending this module to work with any non-DICOM data would be difficult
 - however, this script is still a useful code skeleton for people with their own non-DICOM data
 - the python script hard codes a number of things: 1. there is only 1 project, 1 session, 1 scan, 1 resource, and 1 file associated with a subject 2. metadata is generated with a separate R script, and must conform to a specific schema in order to qualify as a valid upload
 - all of the QA is specific to MICE
- a log file is produced in a subdirectory called 'logs' every time the script is run, allowing one to quickly see the actual files that were uploaded
- the bottleneck of this script is the actual uploading of the files because some of the files are up to 500 MB
 - currently it takes about 10-15 seconds to upload a 75 MB file
 - future development could focus on speeding up this upload somehow (e.g. compression)

TO DO

- delete a subject/session/scan prior to PUT?
 - what if an entity has extra content that it should not have?
 - what if the subject can't be deleted for some reason? this has happened to me before, producing an HTTP 403 error
- verify that the upload was successful (both the file and metadata are identical to the intended upload)
 - how to check that metadata are accurate?
 - * could always manually verify the metadata
 - * could GET the data and check it against what was added
 - how to check that the MINC file is accurate?
 - * is there information contained in the HTTP response body that I can use?
 - * use a checksum somehow?
 - * performing a GET and checking would be somewhat expensive

- * manually spot check certain files with a trained eye?
- extend script to work for reconstructed / pipeline processed images
 - most places would download the MINC files, convert to NIfTI then analyze with FSL