Heudiconv

...

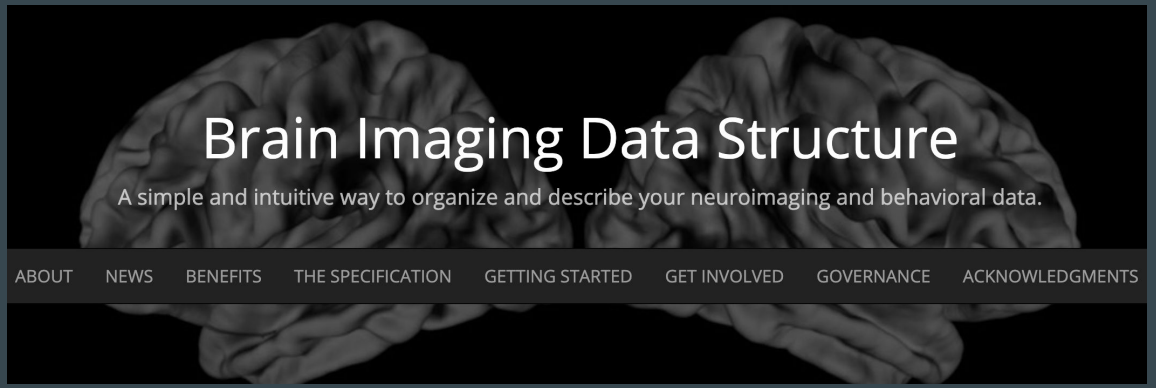
The dicom-to-nifti pipeline

Brown University Clinical
Neuroimaging Research Core

Version Date 01/13/2022

What is Heudiconv?

- A Python-based *heuristic* for *converting* dicom into nifti files and creating a structured data hierarchy like BIDS.
- Easiest to use containerized versions of Heudiconv
  - Mac – Docker
  - Linux – Singularity



https://bids.neuroimaging.io/

BIDS is a *community* developed standardized data structure that makes organizing and sharing data easier

What is a container?

- Software containers group the computer code and all software dependencies needed to run an application or pipeline into one tidy package.
- Easier to use than “bare metal” implementations that require user to install and link multiple dependencies.

https://www.docker.com/products/docker-hub

https://sylabs.io/guides/3.0/user-guide/build_a_container.html

Heudiconv on CCV

Prepping to use containers -- the cache directory

- Apps produce intermediate files that are stored in a temporary buffer or *cache directory*.
- You must *set up* your cache directory *before running* a Singularity container *on CCV*
- To do this, use a *text editor* to modify your *.bashrc* file

- What is a .bashrc file?
  - Hidden file in your home directory setting the default behavior, directories needed for bash to function

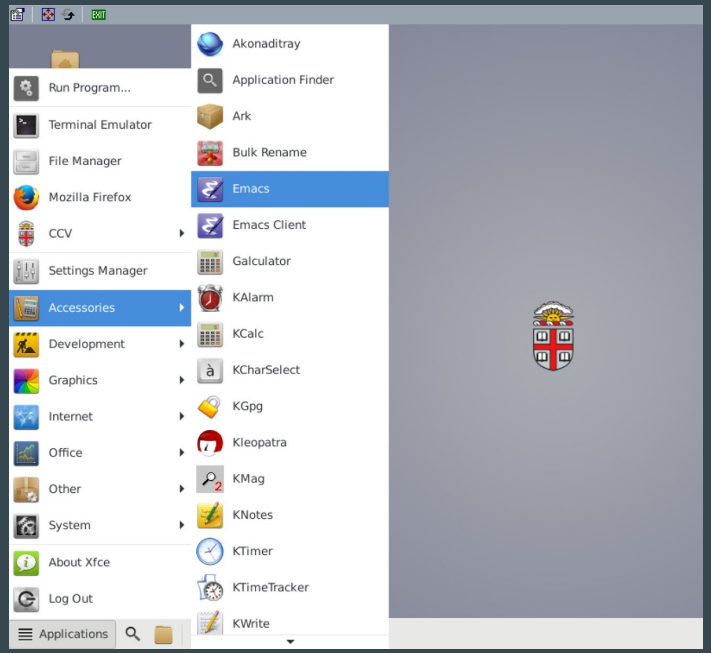
How do I load and open a text editor on CCV?

Through terminal:

- Load a text editing module
- Run by typing the module name
  - Add an ampersand to run “in the background”
  - Means you can still used the command line even though another module is running.

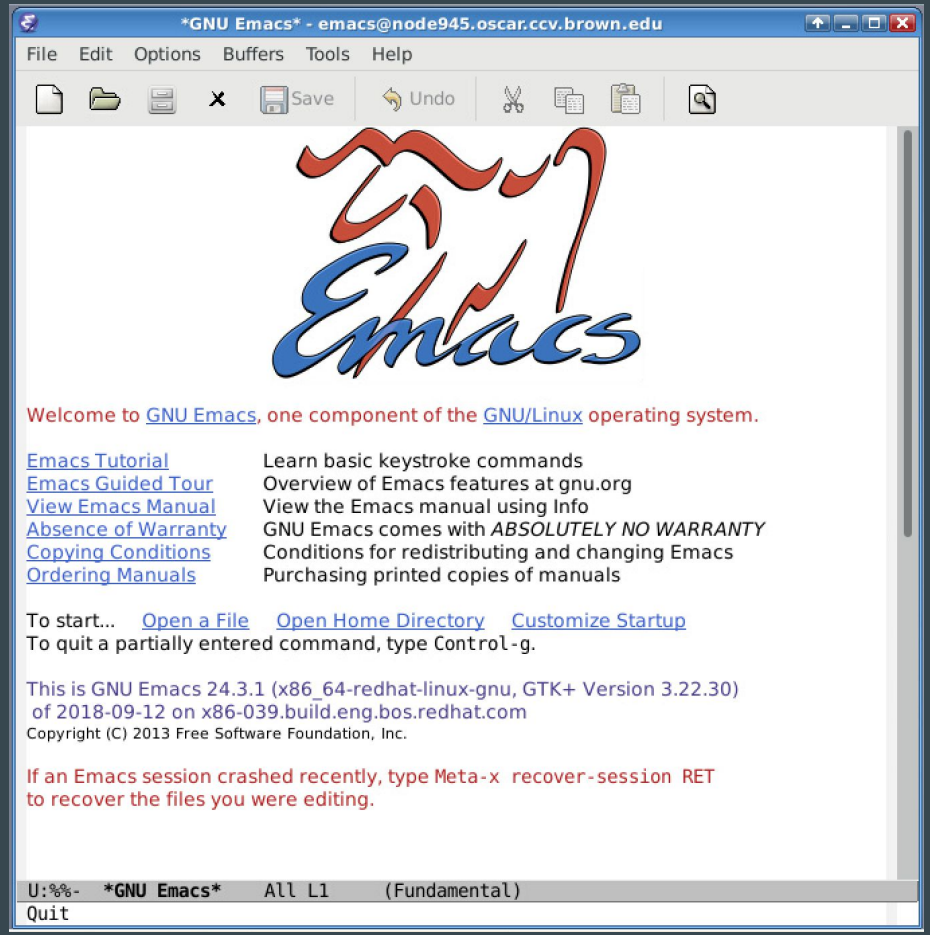
```
[[jbarredo@login005 ~]$ module load emacs/26.3
 module: loading 'emacs/26.3'
[[jbarredo@login005 ~]$ emacs&
```

Through VNC:

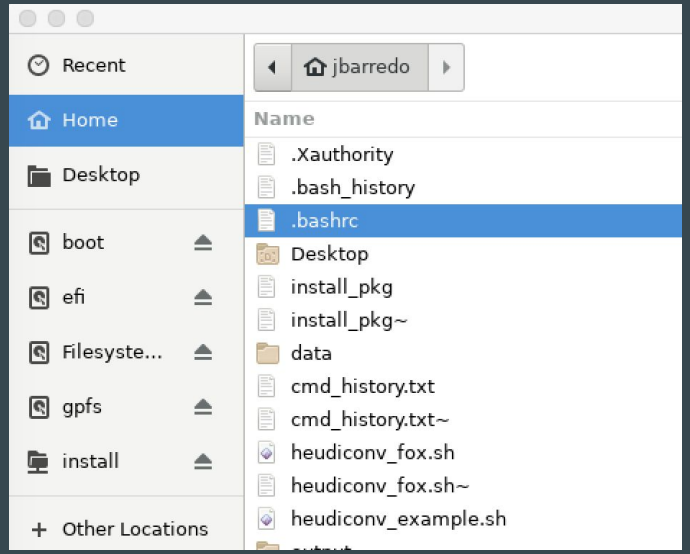


Text Editor: Emacs

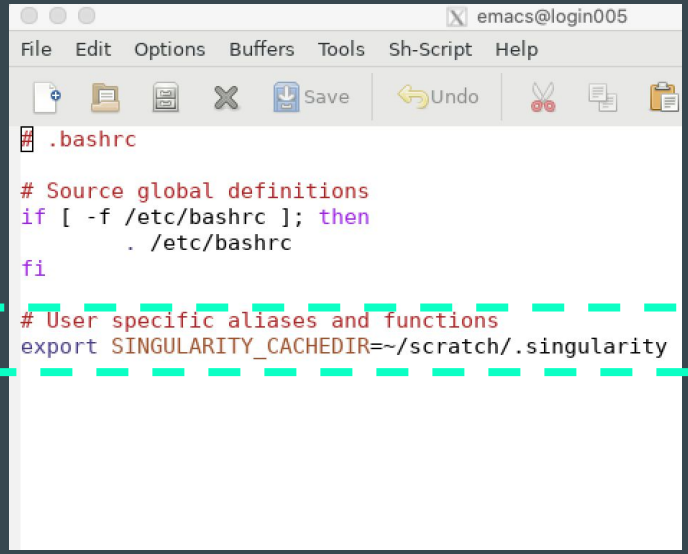
- When you open your text editor it will look like this
- Click File in the top left hand corner, then Visit New File in order to search for the .bashrc file



Modifying .bashrc



After clicking Visit New File
Search .bashrc and then hit enter --
which will open the .bashrc file



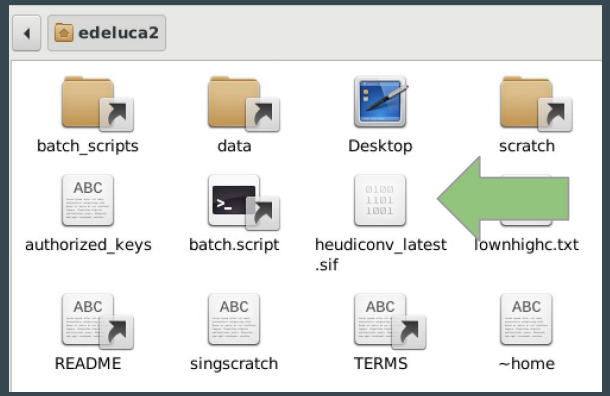
Open .bashrc, add the highlighted lines
and 'save' and then close the file
You will not need to repeat this step
again when running heudiconv

Downloading a new heudiconv container

1. Login to CCV from your desktop terminal or open your terminal through the VNC
2. Go to your home directory, then type:

   singularity pull docker://nipy/heudiconv

This will make a singularity download called heudiconv_latest.sif in your home directory
Once you have downloaded this you will not need to repeat this step again when running heudiconv



Step 1 – Generate an initial heuristic file

- **Heudiconv** is a Python script that describes how to translate dicom images into nifti format
- It's customizable – you do not have to convert all dicoms into nifti format (e.g., auto-align scouts, localizers)
- You must first generate a list of all types of dicom files you have before deciding which to convert and which to ignore
- The steps to generate this list (in a .tsv file) are detailed in the following slides

Step 1 – Generate an initial heuristic file

1a. Start an interactive session from your command line:

```
>> interact -n 1 -t 02:00:00 -m 4g
```

Steps 1b and 1c (next slide) should be entered into a text editor (ex. emacs) and then copied into the command line after modifying -- see slide 12 for further instructions

1b. Define variables pointing to the ***container download*** and the **study's data directory:**
You will need to re-enter these into your terminal every time you open a new one

```
>> img=/gpfs/home/<your username>/heudiconv_latest.sif

>> studydir=/gpfs/data/<your PI's username>/<directory dicoms are in>
```

Step 1 – Generate an initial heuristic file

1c. Set modifier flags and run heudiconv through the singularity container:

```
>> singularity run --cleanenv \
        -B ${studydir}:/base \
        -e ${img} \
        -d /base/sourcedata/sub-{subject}/ses-{session}/*/*.IMA \
        -o /base/rawdata \
        -f convertall \
        -c none \
        -s 2063 \
        -ss 01
```

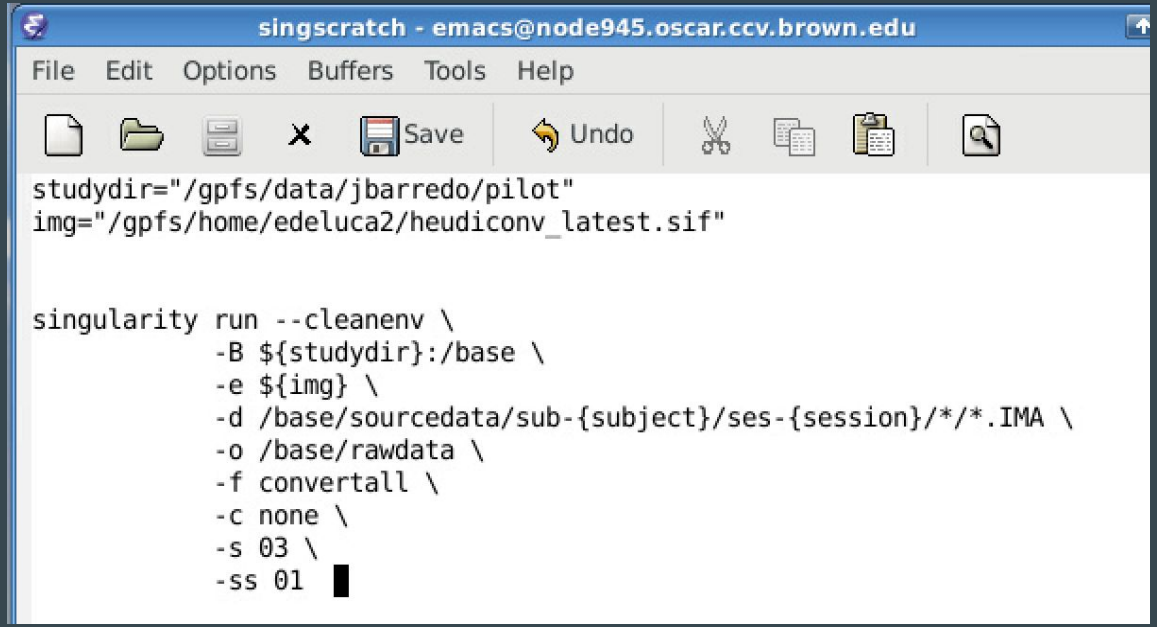
Critical concept -- ‘binding’ to a container. Singularity images are self-contained units. They only know what you tell them. When you bind something, you are mapping directories and permissions to the container. It’s like mounting a drive.

Modifiers

-B *binds your study directory to heudiconv ‘base’ directory*
-e *the container*
-d *dicom directory*
-o *output directory*
-f *python script or built-in heuristic*
-c *conversion program*
-s *subject ID*
-ss *session number*
- * a wildcard which denotes that another folder or file of any name exists between the prior folder and the .IMA dicom files

Step 1 - Generate an initial heuristic file

- Open a new text editor and copy the code from the previous slide
- This allows you to copy and paste the code into the command line and more easily edit any errors
- This code will generate the files you need to create your custom heuristic script



Step 2 – Find the hidden output files

Running heudiconv makes a hidden folder in your output directory. To reveal hidden files enter the following on the command line:

```
>> ls -a $studydir/rawdata/.heudiconv/<subID>/info
```

If you leave out the –a, the file will be invisible. SubID is written out as just the number: 01 not sub-01

You should see the following files:
01_ses-01.auto.txt dicominfo_ses-01.tsv 01_ses-01.edit.txt filegroup_ses-01.json heuristic.py

Now, open a second bash terminal window on your computer's desktop and enter:

```
>> scp -r <username>@ssh.ccv.brown.edu:/<full path to study folder>/rawdata/.heudiconv/<subID>/info ~/Desktop
```

This will copy the hidden folder to your personal computer. You will need to open the .tsv with Excel or another spreadsheet program on your desktop computer.

Step 3 - Create a custom heuristic script

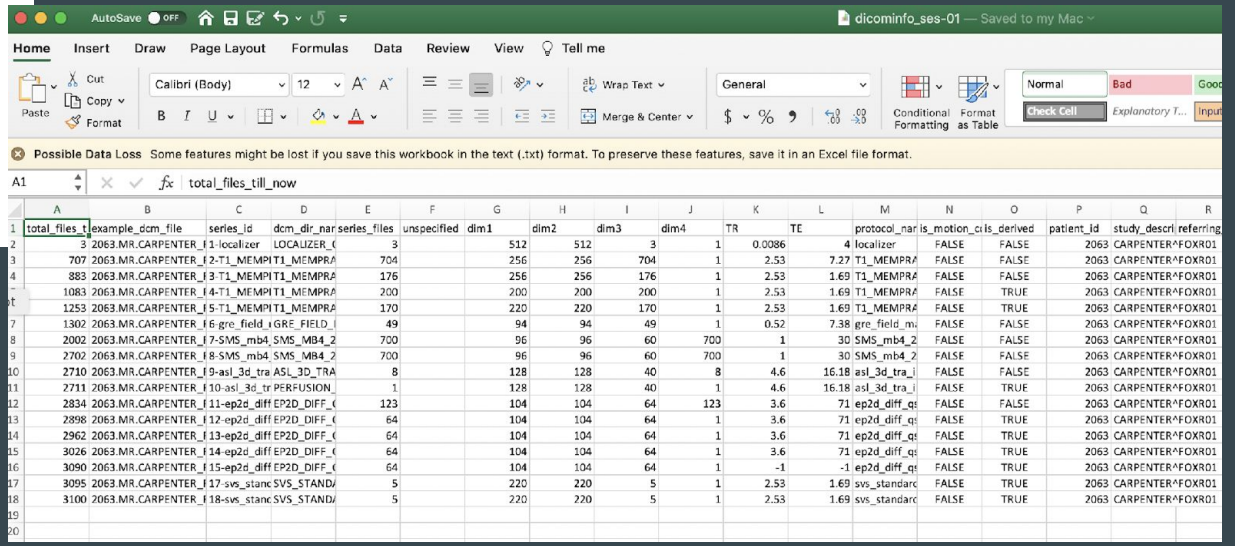
1. Search for, and then open, the info folder on your desktop
2. Open dicominfo.tsv file in Excel
    a. You will not use every dicom/scan
    b. Will most likely exclude any AA head scouts and localizers as these are positional scans
    c. Other dicoms and scans you MAY not use
        - The T1_MEMPRAGE that does not end in RMS under the series_description
        - A Diffusion scan that is listed as Derived under image_type
3. Open heuristic.py in a text editor (ex. aquamacs for mac users)
    a. You will edit this script using the contents of the dicominfo.tsv file to create your custom heuristic file
4. See Next slide for images of each of these files

Step 3 – Create a custom heuristic script

```python
import os


def create_key(template, outttype=('nii.gz',), annotation_classes=None):
    if template is None or not template:
        raise ValueError('Template must be a valid format string')
    return template, outttype, annotation_classes


def infotodict(seqinfo):
    """Heuristic evaluator for determining which runs belong where

    allowed template fields - follow python string module:

    item: index within category
    subject: participant id
    seqitem: run number during scanning
    subindex: sub index within group
    """

    data = create_key('run{item:03d}')
    info = {data: []}
    last_run = len(seqinfo)
```

Modifiable Python script that applies the heuristic

The .tsv file lists dicom fields in the row header. Subsequent rows correspond to runs of the MRI scanner. Unique values or combinations are used to distinguish different runs.



total_files_t	example_dcm_file	series_id	dcm_dir_nar	series_files	unspecified	dim1	dim2	dim3	dim4	TR	TE	protocol_nar	is_motion_c	is_derived	patient_id	study_descri	referring
3	2063.MR.CARPENTER_[	1-localizer	LOCALIZER_(	3		512	512	3	1	0.0086	4	localizer	FALSE	FALSE	2063	CARPENTER^FOXR01	
707	2063.MR.CARPENTER_[	2-T1_MEMPI	T1_MEMPRA	704		256	256	704	1	2.53	7.27	T1_MEMPRA	FALSE	FALSE	2063	CARPENTER^FOXR01	
883	2063.MR.CARPENTER_[	3-T1_MEMPI	T1_MEMPRA	176		256	256	176	1	2.53	1.69	T1_MEMPRA	FALSE	FALSE	2063	CARPENTER^FOXR01	
1083	2063.MR.CARPENTER_[	4-T1_MEMPI	T1_MEMPRA	200		200	200	200	1	2.53	1.69	T1_MEMPRA	FALSE	TRUE	2063	CARPENTER^FOXR01	
1253	2063.MR.CARPENTER_[	5-T1_MEMPI	T1_MEMPRA	170		220	220	170	1	2.53	1.69	T1_MEMPRA	FALSE	TRUE	2063	CARPENTER^FOXR01	
1302	2063.MR.CARPENTER_[	6-gre_field_	GRE_FIELD_(	49		94	94	49	1	0.52	7.38	gre_field_m	FALSE	FALSE	2063	CARPENTER^FOXR01	
2002	2063.MR.CARPENTER_[	7-SMS_mb4	SMS_MB4_2	700		96	96	60	700	1	30	SMS_mb4_2	FALSE	FALSE	2063	CARPENTER^FOXR01	
2702	2063.MR.CARPENTER_[	8-SMS_mb4	SMS_MB4_2	700		96	96	60	700	1	30	SMS_mb4_2	FALSE	FALSE	2063	CARPENTER^FOXR01	
2710	2063.MR.CARPENTER_[	9-asl_3d_tra	ASL_3D_TRA	8		128	128	40	8	4.6	16.18	asl_3d_tra_i	FALSE	FALSE	2063	CARPENTER^FOXR01	
2711	2063.MR.CARPENTER_[	10-asl_3d_tr	PERFUSION_	1		128	128	40	1	4.6	16.18	asl_3d_tra_i	FALSE	TRUE	2063	CARPENTER^FOXR01	
2834	2063.MR.CARPENTER_[	11-ep2d_diff	EP2D_DIFF_(	123		104	104	64	123	3.6	71	ep2d_diff_qs	FALSE	FALSE	2063	CARPENTER^FOXR01	
2898	2063.MR.CARPENTER_[	12-ep2d_diff	EP2D_DIFF_(	64		104	104	64	1	3.6	71	ep2d_diff_qs	FALSE	TRUE	2063	CARPENTER^FOXR01	
2962	2063.MR.CARPENTER_[	13-ep2d_diff	EP2D_DIFF_(	64		104	104	64	1	3.6	71	ep2d_diff_qs	FALSE	TRUE	2063	CARPENTER^FOXR01	
3026	2063.MR.CARPENTER_[	14-ep2d_diff	EP2D_DIFF_(	64		104	104	64	1	3.6	71	ep2d_diff_qs	FALSE	TRUE	2063	CARPENTER^FOXR01	
3090	2063.MR.CARPENTER_[	15-ep2d_diff	EP2D_DIFF_(	64		104	104	64	1	-1	-1	ep2d_diff_qs	FALSE	TRUE	2063	CARPENTER^FOXR01	
3095	2063.MR.CARPENTER_[	17-svs_stanc	SVS_STANDA	5		220	220	5	1	2.53	1.69	svs_standard	FALSE	TRUE	2063	CARPENTER^FOXR01	
3100	2063.MR.CARPENTER_[	18-svs_stanc	SVS_STANDA	5		220	220	5	1	2.53	1.69	svs_standard	FALSE	TRUE	2063	CARPENTER^FOXR01	

Step 3 – Create a custom heuristic script

1. The first section “def create_key” -> leave as is
2. Second section “def infotodict” -> uses the create_key function to define nifti file names and locations for each scanner run according to BIDS.
   a. You will need to follow bids specification for naming
   b. The variable name (left of the =) can be whatever you want
   c. You are telling heudiconv how you want your nifti scan files named and stored
3. Following the example on the following page and the dicom.tsv file you generated you will be making a variable for each scan you wish to convert from dicom to nifti
4. See following slide for relevant images to model your heuristic after

Step 3 – Create a custom heuristic script

Dummy script

```
import os


def create_key(template, outtype=('nii.gz',), annotation_classes=None):
    if template is None or not template:
        raise ValueError('Template must be a valid format string')
    return template, outtype, annotation_classes


def infotodict(seqinfo):
    """Heuristic eval[illegible]termining which runs belong where

    allowed tem[illegible]lds - fo[illegible]ython string module:

    item: inde[illegible]thin [illegible]ry
    subject: p[illegible]cipant [illegible]
    seqitem: ru[illegible]ber duri[illegible]g
    subindex: su[illegible] within
    """

    data = create_key('run{item:03d}')
    info = {data: []}
    last_run = len(seqinfo)
```

Edited script

```
def infotodict(seqinfo):
    # Key definitions. Revised for each new study or after modifying the study design.
    ###################################################################
    # Use the create_key function to define a unique variable for each sequence:
    # variable = create_key(output_directory_path_and_name).
    data = create_key('run-{item:d}')
    t1w = create_key('sub-{subject}/{session}/anat/sub-{subject}_{session}_T1w')
    rest1 = create_key('sub-{subject}/{session}/func/sub-{subject}_{session}_task-rest_run-1_bold')
    rest2 = create_key('sub-{subject}/{session}/func/sub-{subject}_{session}_task-rest_run-2_bold')
    asl = create_key('sub-{subject}/{session}/asl/sub-{subject}_{session}_asl')
    diff = create_key('sub-{subject}/{session}/dwi/sub-{subject}_{session}_dwi')
    mrs1 = create_key('sub-{subject}/{session}/mrs/sub-{subject}_{session}_satoff_mrs')
    mrs2 = create_key('sub-{subject}/{session}/mrs/sub-{subject}_{session}_rdacc_mrs')

    # This data dictionary containing user-defined variables also subject to revision.
    ###################################################################
    # Enter a key in the dictionary for each key you created above in section 1.
    info = {data: [], t1w: [], rest1: [], rest2: [], asl: [], diff: [], mrs1: [], mrs2: []}
    last_run = len(seqinfo)
```

The key defines nifti file names and storage locations and should follow BIDS:

https://bids-specification.readthedocs.io/en/stable/04-modality-specific-files/01-magnetic-resonance-imaging-data.html

Step 3 – Create a custom heuristic script

1. The info variable creates an empty field for each key.
   a. These fields are used to sort dicoms as they are matched to keys
2. You will create a key for each variable you defined in the prior section
3. If you named one of your variables t1w it must match up with a t1w key in this section

```
# This data dictionary containing user-defined variables also subject to revision.
################################################################################
# Enter a key in the dictionary for each key you created above in section 1.
info = {data: [], t1w: [], rest1: [], rest2: [], asl: [], diff: [], mrs1: [], mrs2: []}
last_run = len(seqinfo)
```

Step 3 – Create a custom heuristic script

```
for s in seqinfo:
    """
    The namedtuple `s` contains the following fields:

    * total_files_till_now
    * example_dcm_file
    * series_id
    * dcm_dir_name
    * unspecified2
    * unspecified3
    * dim1
    * dim2
    * dim3
    * dim4
    * TR
    * TE
    * protocol_name
    * is_motion_corrected
    * is_derived
    * patient_id
    * study_description
    * referring_physician_name
    * series_description
    * image_type
    """

    info[data].append(s.series_id)
return info
```

```
# Section 2: These criteria should be revised by user.
########################################################
# Define test criteria to check that each dicom sequence is classified correctly.
# seqinfo (s) refers to information in dicominfo.tsv. Consult that file for available criteria.
# Here we use two types of criteria:
# 1) An equivalent field "==" (e.g., good for checking dimensions)
# 2) A field that includes a string (e.g., 'mprage' in s.protocol_name)

for idx, s in enumerate(seqinfo):
    if ('T1_MEMPRAGE_1.0mm_p2 RMS' in s.series_description):
        info[t1w].append(s.series_id)
    if ('7-SMS_mb4_2mm_TR1000REST' in s.series_id):
        info[rest1].append(s.series_id)
    if ('8-SMS_mb4_2mm_TR1000REST' in s.series_id):
        info[rest2].append(s.series_id)
    if (s.dim4 == 8) and ('*tgse3d1_10080' in s.sequence_name):
        info[asl].append(s.series_id)
    if ('ep2d_diff_qspace_p2_s2' in s.series_description):
        info[diff].append(s.series_id)
    if ('svs_standard_GLX_watersat_off' in s.dcm_dir_name):
        info[mrs1].append(s.series_id)
    if ('svs_standard_GLX_RDACC' in s.dcm_dir_name):
        info[mrs2].append(s.series_id)
return info
```

See the following slide for more detailed instructions

Step 3 – Create a custom heuristic script

1. Erase the section crossed out on the previous page.
2. Create an **if** statement for each key.
   - This defines combinations of header values distinguishing different data runs
   - This information is in the .tsv file
3. For example, if only one scan has the series description value ‘T1_MEMPRAGE_1.0mm_p2 RMS,’ the series description field can be used to tag MEMPRAGE dicoms:

Header value

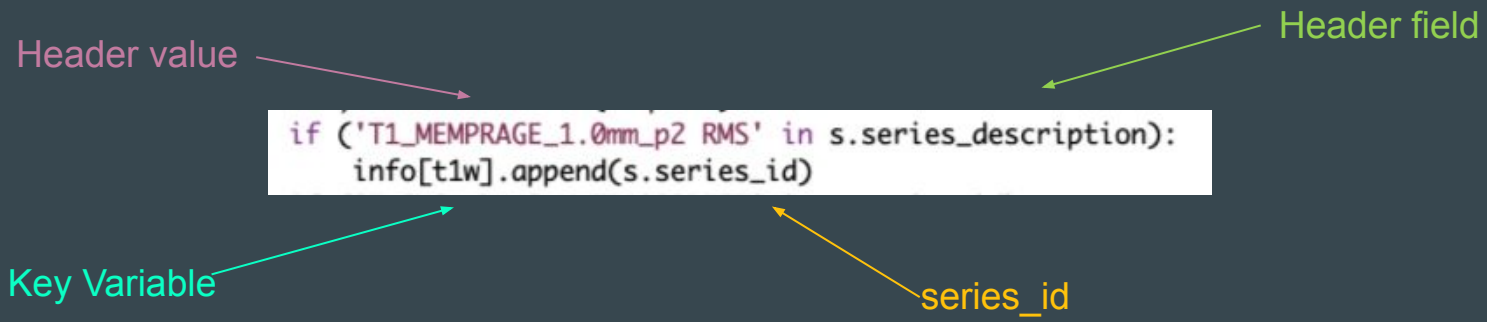
Header field

```
if ('T1_MEMPRAGE_1.0mm_p2 RMS' in s.series_description):
    info[t1w].append(s.series_id)
```

The part to the right of the colon matches dicom series_id values to key variables (‘t1w’).
See next slide for more information

Step 3 – Create a custom heuristic script

- The Header value, Header field, and Key variable will be specific to each dicom/scan you are converting
  - You will get the Header value and the Header field from the .tsv file
  - If you pick the unique identifier for your Header value from the series_description section then your Header field will be series_description
  - Your Key variable must match the define variable and key section from the first half of the heuristic script
- s.series _id always stays the same



Step 4 – Transfer new heuristic script to server and run

Save the new heuristic as heuristic_revised.py and copy it to your code folder on CCV:

```
>> scp ~/Desktop/info/heuristic_revised.py <username>@ssh.ccv.brown.edu:/<full path to study folder>/code
```

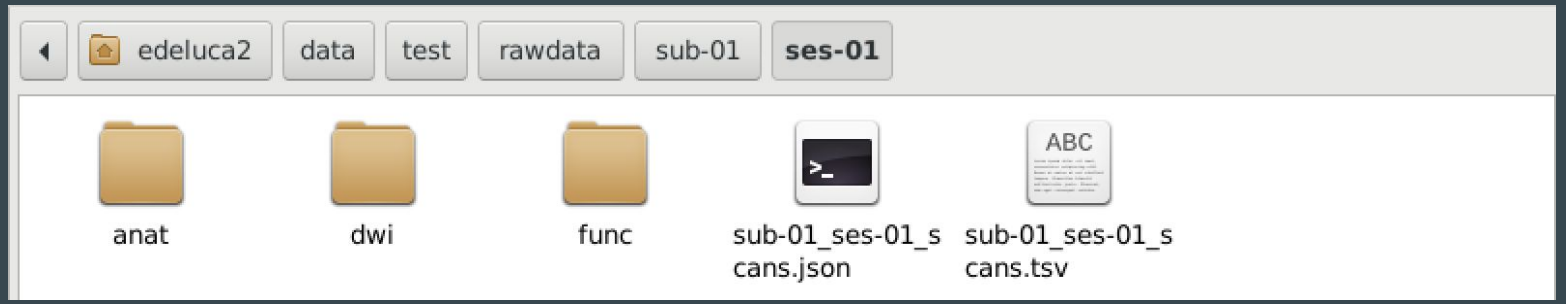
Enter the following code to run Heudiconv:

Green text indicates changes from the initial code on Slide 10. Add this code to the text file you created earlier

```
>> singularity run --cleanenv \
        -B ${studydir}:/base \
        -e ${img} \
        -d /base/sourcedata/sub-{subject}/ses-{session}/*/*.IMA \
        -o /base/rawdata \
        -f /base/code/heuristic.py \
        -c dcm2niix \
        -s 2063 \
        -ss 01 \
        --minmeta \
        -b
```

-**f** *python script or built-in heuristic*
-**c** *conversion program*
-**s** *subject ID*
-**ss** *session number*
-**b** *output in BIDS format*
--**minmeta** includes only required metadata in the JSON file
--**overwrite** *overwrites old results (may need to add as a flag if heudiconv says some files already exist)*

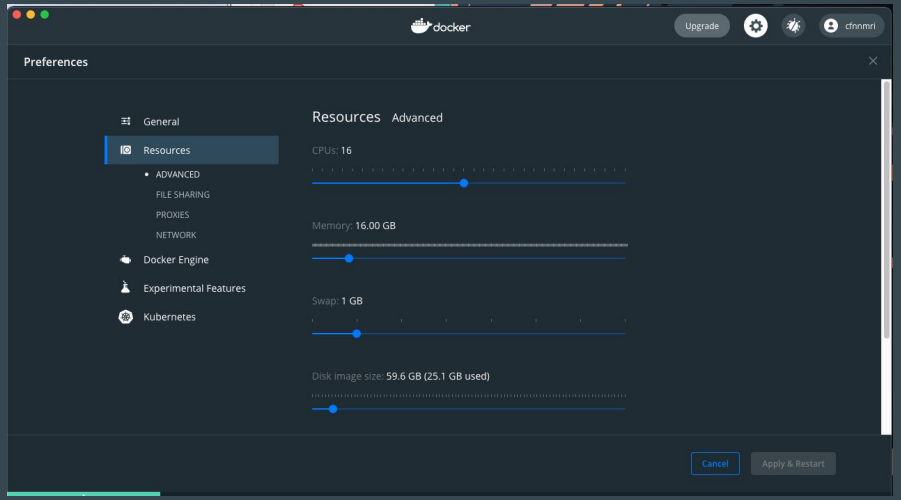
Successful Output Directory Example



Heudiconv on local computer - using docker

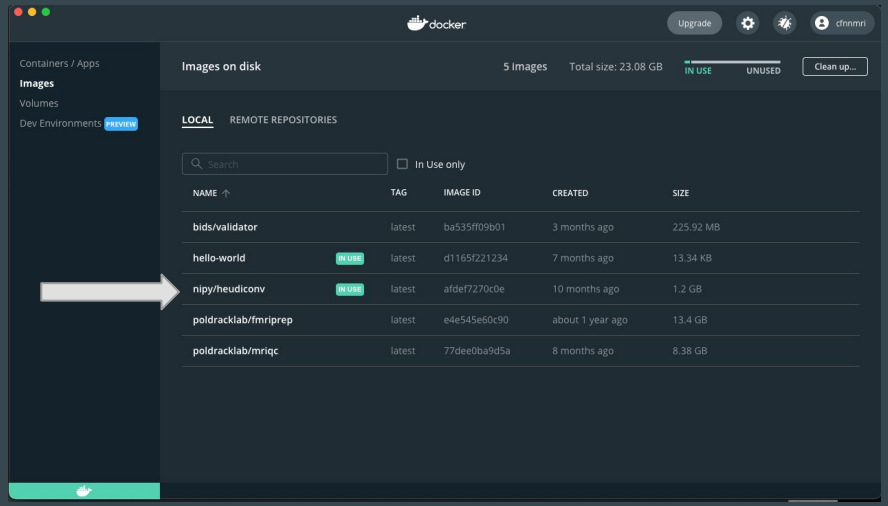
Docker install and set-up

- Install docker on your computer -> download link here
  - You may need administrator privileges to allow Docker certain permissions
- Once docker installs, open the application
  - You may be prompted to go through a tutorial
- In settings -> Resources
  - Change Memory to be 16.00 GB



Add heudiconv to docker

- To add heudiconv to docker, open your terminal
- You need to pull the latest heudiconv image to your docker application
- In the terminal, type < docker pull nipy/heudiconv >
- After heudiconv is successfully added into your docker application, you should see it look like this:



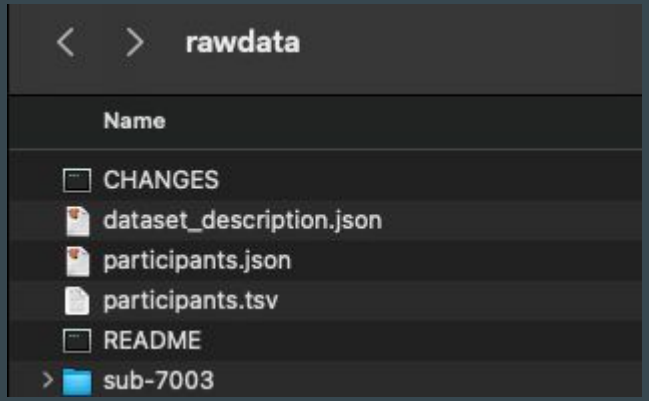
Running heudiconv locally

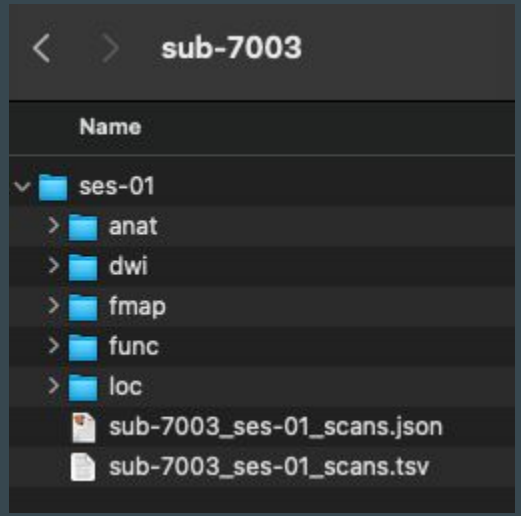
- Similarly to running heudiconv on CCV, you will need to create a heuristic.py file
- First, run this code to generate the dicominfo.tsv file in the hidden directory
  - < docker run --rm -it -v /path/to/study/directory:/base nipy/heudiconv:latest -d /base/sourcedata/sub-{subject}/ses-{session}/*/*/*/* -o /base/rawdata/ -f convertall -s 01 -ss 01 -c none --overwrite >
  - Please review above slides for the exact flag usage
- Locate the hidden directory by typing < ls -a $studydir/rawdata/.heudiconv/<subID>/info >
- Open the info folder and locate the dicominfo.tsv file
- Open the dicominfo.tsv file and create your heuristic.py file and save it in your code directory (see slides above for more information on creating the heuristic file)
- Save your new heuristic.py file in the $studydir/code directory

Running heudiconv locally

- Now that you have your newly revised heuristic file, we can run heudiconv and convert our dicoms
- You should be in your $studydir/code directory to run this command
- While docker is open and running, type the following command in your terminal:
  - < docker run --rm -it -v /path/to/study/directory:/base nipy/heudiconv:latest -d /base/sourcedata/sub-{subject}/ses-{session}/*/*/*/* -o /base/rawdata/ -f heuristic.py -s 01 -ss 01 -c dcm2niix --overwrite >
  - The -f and -c flags are the only flags changing from the original command
- Note: after you have created your heuristic file, you don't need to create a new heuristic file each time you need to run heudiconv **unless there is a change in the data you have collected**

Successful output example





Inside each of these folders will be NiFti and JSON files for each of your sequences.