heudiconv Documentation

Release 0.10.0

Heudiconv team

Feb 15, 2023

Contents

a heuristic-centric DICOM converter

CHAPTER 1

About

`heudiconv` is a flexible DICOM converter for organizing brain imaging data into structured directory layouts.

- it allows flexible directory layouts and naming schemes through customizable heuristics implementations
- it only converts the necessary DICOMs, not everything in a directory
- you can keep links to DICOM files in the participant layout
- using dcm2niix under the hood, it's fast
- it can track the provenance of the conversion from DICOM to NIfTI in W3C PROV format
- it provides assistance in converting to BIDS.
- it integrates with DataLad to place converted and original data under git/git-annex version control, while automatically annotating files with sensitive information (e.g., non-defaced anatomicals, etc)

CHAPTER 2

How to cite

Please use Zenodo record for your specific version of HeuDiConv. We also support gathering all relevant citations via DueCredit.

CHAPTER 3

Contents

3.1 Installation

`Heudiconv` is packaged and available from many different sources.

3.1.1 Local

Released versions of HeuDiConv are available on PyPI and conda. If installing through `PyPI`, eg:

```
pip install heudiconv[all]
```

Manual installation of dcm2niix is required.

On Debian-based systems we recommend using NeuroDebian which provides the heudiconv package.

3.1.2 Docker

If Docker is available on your system, you can visit our page on Docker Hub to view available releases. To pull the latest release, run:

```
$ docker pull nipy/heudiconv:0.10.0
```

3.1.3 Singularity

If Singularity is available on your system, you can use it to pull and convert our Docker images! For example, to pull and build the latest release, you can run:

```
$ singularity pull docker://nipy/heudiconv:0.10.0
```

3.2 Changes

3.2.1 Changelog

All notable changes to this project will be documented (for humans) in this file.

The format is based on Keep a Changelog and this project adheres to Semantic Versioning.

[0.10.0] - 2021-09-16

Various improvements and compatibility/support (dcm2niix, datalad) changes.

Added

- Add "AcquisitionTime" to the seqinfo (#487)
- Add support for saving the Phoenix Report in the sourcedata folder (#489)

Changed

- Python 3.5 EOLed, supported (tested) versions now: 3.6 - 3.9
- In reprorin heuristic, allow for having multiple accessions since now there is `-g all` groupping (#508)
- For BIDS, produce a singular `scans.json` at the top level, and not one per sub/ses (generates too many identical files) (#507)

Fixed

- Compatibility with DataLad 0.15.0. Minimal version is 0.13.0 now.
- Try to open top level BIDS .json files a number of times for adjustment, so in the case of competition across parallel processes, they just end up with the last one "winning over" (#523)
- Don't fail if etelemetry.get_project returns None (#501)
- Consistently use `n/a` for age/sex, also handle ?M for months (#500)
- To avoid crashing on unrelated derivatives files etc, make `find_files` to take list of topdirs (excluding `derivatives/` etc), and look for _bold only under sub-* directories (#496)
- Ensure bvec/bval files are only created for dwi output (#491)

Removed

- In reproin heuristic, old hardcoded sequence renamings and filters (#508)

[0.9.0] - 2020-12-23

Various improvements and compatibility/support (dcm2niix, datalad, duecredit) changes. Major change is placement of output files to the target output directory during conversion.

Added

- #454 zenodo referencing in README.rst and support for ducredit for heudiconv and reproin heuristic
- #445 more tutorial references in README.md

Changed

- #485 placed files during conversion right away into the target directory (with a `_heudiconv???` suffix, renamed into ultimate target name later on), which avoids hitting file size limits of /tmp (#481) and helped to avoid a regression in dcm2nixx 1.0.20201102
- #477 replaced `rec-<magnitude|phase>` with `part-<mag|phase>` now hat BIDSsupports the part entity
- #473 made default for CogAtlasID to be a TODO URL
- #459 made AcquisitionTime used for acq_time scans file field
- #451 retained sub-second resolution in scans files
- #442 refactored code so there is now heudiconv.main.workflow for more convenient use as a Python module

Fixed

- minimal version of nipype set to 1.2.3 to guarantee correct handling of DWI files (#480)
- `heudiconvDCM*` temporary directories are removed now (#462)
- compatibility with DataLad 0.13 (#464)

Removed

- #443 pathlib as a dependency (we are Python3 only now)

[0.8.0] - 2020-04-15

Enhancements

- Centralized saving of .json files. Indentation of some files could change now from previous versions where it could have used `3` spaces. Now indentation should be consistently `2` for .json files we produce/modify (#436) (note: dcm2niix uses tabs for indentation)
- ReproIn heuristic: support SBRef and phase data (#387)
- Set the "TaskName" field in .json sidecar files for multi-echo data (#420)
- Provide an informative exception if command needs heuristic to be specified (#437)

Refactored

- `embed_nifti` was refactored into `embed_dicom_and_nifti_metadata` which would no longer create `.nii` file if it does not exist already (#432)

Fixed

- Skip datalad-based tests if no datalad available (#430)
- Search heuristic file path first so we do not pick up a python module if name conflicts (#434)

[0.7.0] - 2020-03-20

Removed

- Python 2 support/testing

Enhancement

- `-g` option obtained two new modes: `all` and `custom`. In case of `all`, all provided DICOMs will be treated as coming from a single scanning session. `custom` instructs to use `.grouping` value (could be a DICOM attribute or a callable)provided by the heuristic (#359).
- Stop before reading pixels data while gathering metadata from DICOMs (#404)
- reproin heuristic:
  - In addition to original "md5sum of the study_description" `protocols2fix` could now have (and applied after md5sum matching ones) 1). a regular expression searched in study_description, 2). an empty string as "catch all". This features could be used to easily provide remapping into reproin naming (documentation is to come to http://github.com/ReproNim/reproin) (#425)

Fixed

- Use nan, not None for absent echo value in sorting
- reproin heuristic: case seqinfos into a list to be able to modify from overloaded heuristic (#419)
- No spurious errors from the logger upon a warning about `etelemetry` absence (#407)

[0.6.0] - 2019-12-16

This is largely a bug fix. Metadata and order of `_key-value` fields in BIDS could change from the result of converting using previous versions, thus minor version boost. 14 people contributed to this release – thanks everyone!

Enhancement

- Use etelemetry to inform about most recent available version of heudiconv. Please set `NO_ET` environment variable if you want to disable it (#369)
- BIDS:
  - `--bids` flag became an option. It can (optionally) accept `notop` value to avoid creation of top level files (`CHANGES`, `dataset_description.json`, etc) as a workaround during parallel execution to avoid race conditions etc. (#344)
  - Generate basic `.json` files with descriptions of the fields for `participants.tsv` and `_scans.tsv` files (#376)

  - Use `filelock` while writing top level files. Use `HEUDICONV_FILELOCK_TIMEOUT` environment to change the default timeout value (#348)
  - `_PDT2` was added as a suffix for multi-echo (really "multi-modal") sequences (#345)
- Calls to `dcm2niix` would include full output path to make it easier to discern in the logs what file it is working on (#351)
- With recent **'datalad <>'_** (>= 0.10), created DataLad dataset will use `--fake-dates` functionality of DataLad to not leak data conversion dates, which might be close to actual data acquisition/patient visit (#352)
- Support multi-echo EPI `_phase` data (#373 fixes #368)
- Log location of a bad .json file to ease troubleshooting (#379)
- Add basic pypi classifiers for the package (#380)

Fixed

- Sorting `_scans.tsv` files lacking valid dates field should not cause a crash (#337)
- Multi-echo files detection based number of echos (#339)
- BIDS
  - Use `EchoTimes` from the associated multi-echo files if `EchoNumber` tag is missing (#366 fixes #347)
  - Tolerate empty ContentTime and/or ContentDate in DICOMs (#372) and place "n/a" if value is missing (#390)
  - Do not crash and store original .json file is "JSON pretification" fails (#342)
- ReproIn heuristic
  - tolerate WIP prefix on Philips scanners (#343)
  - allow for use of `(...)` instead of `{...}` since `{}` are not allowed (#343)
  - Support pipolar fieldmaps by providing them with `_epi` not `_magnitude`. "Loose" BIDS `_key-value` pairs might come now after `_dir-` even if they came first before (#358 fixes #357)
- All heuristics saved under `.heudiconv/` under `heuristic.py` name, to avoid discrepancy during reconversion (#354 fixes #353)
- Do not crash (with TypeError) while trying to sort absent file list (#360)
- heudiconv requires nipype >= 1.0.0 (#364) and blacklists `1.2.[12]` (#375)

[0.5.4] - 2019-04-29

This release includes fixes to BIDS multi-echo conversions, the re-implementation of queuing support (currently just SLURM), as well as some bugfixes.

Starting today, we will (finally) push versioned releases to DockerHub. Finally, to more accurately reflect on-going development, the `latest` tag has been renamed to `unstable`.

Added

- Readthedocs documentation (#327)

Changed

- Update Docker dcm2niix to v.1.0.20190410 (#334)
- Allow usage of `--files` with basic heuristics. This requires use of `--subject` flag, and is limited to one subject. (#293)

Deprecated

Fixed

- Improve support for multiple `--queue-args` (#328)
- Fixed an issue where generated BIDS sidecar files were missing additional information - treating all conversions as if the `--minmeta` flag was used (#306)
- Re-enable SLURM queuing support (#304)
- BIDS multi-echo support for EPI + T1 images (#293)
- Correctly handle the case when `outtype` of heuristic has "dicom" before '.nii.gz'. Previously would have lead to absent additional metadata extraction etc (#310)

Removed

- `--sbargs` argument was renamed to `--queue-args` (#304)

Security

[0.5.3] - 2019-01-12

Minor hot bugfix release

Fixed

- Do not shorten spaces in the dates while pretty printing .json

[0.5.2] - 2019-01-04

A variety of bugfixes

Changed

- Reproin heuristic: `__dup` indices would now be assigned incrementally individually per each sequence, so there is a chance to properly treat associate for multi-file (e.g. `fmap`) sequences
- Reproin heuristic: also split StudyDescription by space not only by ^
- `tests/` moved under `heudiconv/tests` to ease maintenance and facilitate testing of an installed heudiconv

- Protocol name will also be accessed from private Siemens csa.tProtocolName header field if not present in public one
- nipype>=0.12.0 is required now

Fixed

- Multiple files produced by dcm2niix are first sorted to guarantee correct order e.g. of magnitude files in fieldmaps, which otherwise resulted in incorrect according to BIDS ordering of them
- Aggregated top level .json files now would contain only the fields with the same values from all scanned files. In prior versions, those files were not regenerated after an initial conversion
- Unicode handling in anonimization scripts

[0.5.1] - 2018-07-05

Bugfix release

Added

- Video tutorial / updated slides
- Helper to set metadata restrictions correctly
- Usage is now shown when run without arguments
- New fields to Seqinfo
  - series_uid
- Reproin heuristic support for xnat ### Changed
- Dockerfile updated to use `dcm2niix v1.0.20180622`
- Conversion table will be regenerated if heurisic has changed
- Do not touch existing BIDS files
  - events.tsv
  - task JSON ### Fixed
- Python 2.7.8 and older installation
- Support for updated packages
  - `Datalad` 0.10
  - `pydicom` 1.0.2
- Later versions of `pydicom` are prioritized first
- JSON pretty print should not remove spaces
- Phasediff fieldmaps behavior
  - ensure phasediff exists
  - support for single magnitude acquisitions

[0.5] - 2018-03-01

The first release after major refactoring:

Changed

- Refactored into a proper `heudiconv` Python module
  - `heuristics` is now a `heudiconv.heuristics` submodule
  - you can specify shipped heuristics by name (e.g. `-f reproin`) without providing full path to their files
  - you need to use `--files` (not just positional argument(s)) if not using `--dicom_dir_templates` or `--subjects` to point to data files or directories with input DICOMs
- `Dockerfile` is generated by neurodocker
- Logging verbosity reduced
- Increased leniency with missing DICOM fields
- `dbic_bids` heuristic renamed into reproin ### Added
- LICENSE with Apache 2.0 license for the project
- CHANGELOG.md
- Regression testing on real data (using datalad)
- A dedicated ReproIn project with details about ReproIn setup/specification and operation using `reproin` heuristic shipped with heudiconv
- utils/test-compare-two-versions.sh helper to compare conversions with two different versions of heudiconv ### Removed
- Support for converters other than `dcm2niix`, which is now the default. Explicitly specify `-c none` to only prepare conversion specification files without performing actual conversion ### Fixed
- Compatibility with Nipype 1.0, PyDicom 1.0, and upcoming DataLad 0.10
- Consistency with converted files permissions
- Ensured subject id for BIDS conversions will be BIDS compliant
- Re-add `seqinfo` fields as column names in generated `dicominfo`
- More robust sanity check of the regex reformatted .json file to avoid numeric precision issues
- Many other various issues

[0.4] - 2017-10-15

A usable release to support DBIC use-case

Added

- more testing ### Changes
- Dockerfile updates (added pigz, progressed forward dcm2niix) ### Fixed
- correct date/time in BIDS `_scans` files

- sort entries in `_scans` by date and then filename

[0.3] - 2017-07-10

A somewhat working release on the way to support DBIC use-case

Added

- more tests
- groupping of dicoms by series if provided
- many more features and fixes

[0.2] - 2016-10-20

An initial release on the way to support DBIC use-case

Added

- basic Python project assets (`setup.py`, etc)
- basic tests
- datalad support
- dbic_bids heuristic
- `--dbg` command line flag to enter `pdb` environment upon failure ## Fixed
- Better Python3 support
- Better PEP8 compliance

[0.1] - 2015-09-23

Initial version

Just a template for future records:

[Unreleased] - Date

TODO Summary

Added

Changed

Deprecated

Fixed

Removed

Security

References

3.3 Usage

`heudiconv` processes DICOM files and converts the output into user defined paths.

3.3.1 CommandLine Arguments

Example: heudiconv -d 'rawdata/{subject}' -o . -f heuristic.py -s s1 s2 s3

```
usage: heudiconv [-h] [--version]
                 [-d DICOM_DIR_TEMPLATE | --files [FILES [FILES ...]]]
                 [-s [SUBJS [SUBJS ...]]] [-c {dcm2niix,none}] [-o OUTDIR]
                 [-l LOCATOR] [-a CONV_OUTDIR] [--anon-cmd ANON_CMD]
                 [-f HEURISTIC] [-p] [-ss SESSION]
                 [-b [BIDSOPTION1 [BIDSOPTION2 ...]]] [--overwrite]
                 [--datalad] [--dbg]
                 [--command {heuristics,heuristic-info,ls,populate-templates,sanitize-
↪jsons,treat-jsons}]
                 [-g {studyUID,accession_number,all,custom}] [--minmeta]
                 [--random-seed RANDOM_SEED] [--dcmconfig DCMCONFIG]
                 [-q {SLURM,None}] [--queue-args QUEUE_ARGS]
```

Named Arguments

--version show program's version number and exit

-d, --dicom_dir_template Location of dicomdir that can be indexed with subject id {subject} and session {session}. Tarballs (can be compressed) are supported in addition to directory. All matching tarballs for a subject are extracted and their content processed in a single pass. If multiple tarballs are found, each is assumed to be a separate session and the –ses argument is ignored. Note that you might need to surround the value with quotes to avoid {...} being considered by shell

--files Files (tarballs, dicoms) or directories containing files to process. Cannot be provided if using –dicom_dir_template.

-s, --subjects	List of subjects - required for dicom template. If not provided, DICOMS would first be "sorted" and subject IDs deduced by the heuristic.
-c, --converter	Possible choices: dcm2niix, none Tool to use for DICOM conversion. Setting to "none" disables the actual conversion step – useful for testing heuristics.
-o, --outdir	Output directory for conversion setup (for further customization and future reference. This directory will refer to non-anonymized subject IDs.
-l, --locator	Study path under outdir. If provided, it overloads the value provided by the heuristic. If –datalad is enabled, every directory within locator becomes a superdataset thus establishing a hierarchy. Setting to "unknown" will skip that dataset.
-a, --conv-outdir	Output directory for converted files. By default this is identical to –outdir. This option is most useful in combination with –anon-cmd.
--anon-cmd	Command to run to convert subject IDs used for DICOMs to anonymized IDs. Such command must take a single argument and return a single anonymized ID. Also see –conv-outdir.
-f, --heuristic	Name of a known heuristic or path to the Python script containing heuristic.
-p, --with-prov	Store additional provenance information. Requires python-rdflib.
-ss, --ses	Session for longitudinal study_sessions. Default is None.
-b, --bids	Possible choices: notop Flag for output into BIDS structure. Can also take BIDS-specific options, e.g., –bids notop. The only currently supported options is "notop", which skips creation of top-level BIDS files. This is useful when running in batch mode to prevent possible race conditions.
--overwrite	Overwrite existing converted files.
--datalad	Store the entire collection as DataLad dataset(s). Small files will be committed directly to git, while large to annex. New version (6) of annex repositories will be used in a "thin" mode so it would look to mortals as just any other regular directory (i.e. no symlinks to under .git/annex). For now just for BIDS mode.
--dbg	Do not catch exceptions and show exception traceback.
--command	Possible choices: heuristics, heuristic-info, ls, populate-templates, sanitize-jsons, treat-jsons Custom action to be performed on provided files instead of regular operation.
-g, --grouping	Possible choices: studyUID, accession_number, all, custom How to group dicoms (default: by studyUID).
--minmeta	Exclude dcmstack meta information in sidecar jsons.
--random-seed	Random seed to initialize RNG.
--dcmconfig	JSON file for additional dcm2niix configuration.

Conversion submission options

-q, --queue	Possible choices: SLURM, None Batch system to submit jobs in parallel.

--queue-args Additional queue arguments passed as a single string of space-separated Argument=Value pairs.

3.3.2 Support

All bugs, concerns and enhancement requests for this software can be submitted here: https://github.com/nipy/heudiconv/issues.

If you have a problem or would like to ask a question about how to use `heudiconv`, please submit a question to NeuroStars.org with a `heudiconv` tag. NeuroStars.org is a platform similar to StackOverflow but dedicated to neuroinformatics.

All previous `heudiconv` questions are available here: http://neurostars.org/tags/heudiconv/

3.3.3 Batch jobs

`heudiconv` can natively handle multi-subject, multi-session conversions, although it will process these linearly. To speed this up, multiple `heudiconv` processes can be spawned concurrently, each converting a different subject and/or session.

The following example uses SLURM and Singularity to submit every subjects' DICOMs as an independent `heudiconv` execution.

The first script aggregates the DICOM directories and submits them to `run_heudiconv.sh` with SLURM as a job array.

If using bids, the `notop` bids option suppresses creation of top-level files in the bids directory (e.g., `dataset_description.json`) to avoid possible race conditions. These files may be generated later with `populate_templates.sh` below (except for `participants.tsv`, which must be create manually).

```
#!/bin/bash

set -eu

# where the DICOMs are located
DCMROOT=/dicom/storage/voice
# where we want to output the data
OUTPUT=/converted/data/voice

# find all DICOM directories that start with "voice"
DCMDIRS=(`find ${DCMROOT} -maxdepth 1 -name voice* -type d`)

# submit to another script as a job array on SLURM
sbatch --array=0-`expr ${#DCMDIRS[@]} - 1` run_heudiconv.sh ${OUTPUT} ${DCMDIRS[@]}
```

The second script processes a DICOM directory with `heudiconv` using the built-in *reproin* heuristic.

```
#!/bin/bash
set -eu

OUTDIR=${1}
# receive all directories, and index them per job array
DCMDIRS=(${@:2})
DCMDIR=${DCMDIRS[${SLURM_ARRAY_TASK_ID}]}
echo Submitted directory: ${DCMDIR}
```

(continues on next page)

(continued from previous page)

```
IMG="/singularity-images/heudiconv-0.10.0-dev.sif"
CMD="singularity run -B ${DCMDIR}:/dicoms:ro -B ${OUTDIR}:/output -e ${IMG} --files /
↪dicoms/ -o /output -f reproin -c dcm2niix -b notop --minmeta -l ."

printf "Command:\n${CMD}\n"
${CMD}
echo "Successful process"
```

This script creates the top-level bids files (e.g., `dataset_description.json`)

..code:: shell #!/bin/bash set -eu

OUTDIR=${1} IMG="/singularity-images/heudiconv-0.10.0-dev.sif" CMD="singularity run -B ${OUTDIR}:/output -e ${IMG} –files /output -f reproin –command populate-templates"

printf "Command:n${CMD}n" ${CMD} echo "Successful process"

3.4 Heuristic

The heuristic file controls how information about the DICOMs is used to convert to a file system layout (e.g., BIDS). `heudiconv` includes some built-in heuristics, including ReproIn (which is great to adopt if you will be starting your data collection!).

However, there is a large variety of data out there, and not all DICOMs will be covered by the existing heuristics. This section will outline what makes up a heuristic file, and some useful functions available when making one.

3.4.1 Components

`infotodict(seqinfos)`

The only required function for a heuristic, *infotodict* is used to both define the conversion outputs and specify the criteria for scan to output association. Conversion outputs are defined as keys, a *tuple* consisting of a template path used for the basis of outputs, as well as a *tuple* of output types. Valid types include *nii*, *nii.gz*, and *dicom*.

Note: An example conversion key

```
('sub-{subject}/func/sub-{subject}_task-test_run-{item}_bold', ('nii.gz',
'dicom'))
```

The `seqinfos` parameter is a list of namedtuples which serves as a grouped and stacked record of the DICOMs passed in. Each item in *seqinfo* contains DICOM metadata that can be used to isolate the series, and assign it to a conversion key.

A dictionary of {`conversion key`: `seqinfo`} is returned.

`create_key(template, outtype)`

A common helper function used to create the conversion key in `infotodict`.

filter_files(fl)

A utility function used to filter any input files.

If this function is included, every file found will go through this filter. Any files where this function returns `True` will be filtered out.

filter_dicom(dcm_data)

A utility function used to filter any DICOMs.

If this function is included, every DICOM found will go through this filter. Any DICOMs where this function returns `True` will be filtered out.

infotoids(seqinfos, outdir)

Further processing on `seqinfos` to deduce/customize subject, session, and locator.

A dictionary of {"locator": locator, "session": session, "subject": subject} is returned.

grouping string or grouping(files, dcmfilter, seqinfo)

Whenever `--grouping custom` (`-g custom`) is used, this attribute or callable will be used to inform how to group the DICOMs into separate groups. From original PR#359:

```
grouping = 'AcquisitionDate'
```

or:

```
def grouping(files, dcmfilter, seqinfo):
    seqinfos = collections.OrderedDict()
    ...
    return seqinfos  # ordered dict containing seqinfo objects: list of DICOMs
```

3.5 User Tutorials

Luckily(?), we live in an era of plentiful information. Below are some links to other users' tutorials covering their experience with `heudiconv`.

- YouTube tutorial by James Kent.
- Walkthrough by the Stanford Center for Reproducible Neuroscience.
- U of A Neuroimaging Core by Dianne Patterson.
- Sample Conversion: Coastal Coding 2019.
- A joined DataLad and HeuDiConv tutorial for reproducible fMRI studies.
- The ReproIn conversion workflow overview.
- Slides and recording of a ReproNim Webinar on `heudiconv`.

Caution: Some of these tutorials may not be up to date with the latest releases of `heudiconv`.

3.6 API Reference

3.6.1 BIDS

Handle BIDS specific operations

exception heudiconv.bids.**BIDSError**

heudiconv.bids.**add_rows_to_scans_keys_file**(*fn*, *newrows*)
Add new rows to file fn for scans key filename and generate accompanying json descriptor to make BIDS validator happy.

fn: filename newrows: extra rows to add

dict fn: [acquisition time, referring physician, random string]

heudiconv.bids.**convert_sid_bids**(*subject_id*)
Strips any non-BIDS compliant characters within subject_id

subject_id : string

sid [string] New subject ID

subject_id [string] Original subject ID

heudiconv.bids.**find_subj_ses**(*f_name*)
Given a path to the bids formatted filename parse out subject/session

heudiconv.bids.**get_formatted_scans_key_row**(*dcm_fn*)
item

row: list [ISO acquisition time, performing physician name, random string]

heudiconv.bids.**maybe_na**(*val*)
Return 'n/a' if non-None value represented as str is not empty

Primarily for the consistent use of lower case 'n/a' so 'N/A' and 'NA' are also treated as 'n/a'

heudiconv.bids.**populate_aggregated_jsons**(*path*)
Aggregate across the entire BIDS dataset .json's into top level .json's

Top level .json files would contain only the fields which are common to all subject[/session]/type/*_modality.json's.

ATM aggregating only for *_task*_bold.json files. Only the task- and OPTIONAL _acq- field is retained within the aggregated filename. The other BIDS _key-value pairs are "aggregated over".

path: str Path to the top of the BIDS dataset

heudiconv.bids.**populate_bids_templates**(*path*, *defaults={}*)
Premake BIDS text files with templates

heudiconv.bids.**save_scans_key**(*item*, *bids_files*)
item: bids_files: str or list

heudiconv.bids.**treat_age**(*age*)
Age might encounter 'Y' suffix or be a float

heudiconv.bids.**tuneup_bids_json_files**(*json_files*)
Given a list of BIDS .json files, e.g.

3.6.2 Conversion

heudiconv.convert.**add_taskname_to_infofile**(*infofiles*)
Add the "TaskName" field to json files corresponding to func images.

infofiles : list with json filenames or single filename

heudiconv.convert.**bvals_are_zero**(*bval_file*)
Checks if all entries in a bvals file are zero (or 5, for Siemens files). Returns True if that is the case, otherwise returns False

bval_file : file with the bvals

True if all are zero; False otherwise.

heudiconv.convert.**convert**(*items*, *converter*, *scaninfo_suffix*, *custom_callable*, *with_prov*, *bids_options*, *outdir*, *min_meta*, *overwrite*, *symlink=True*, *prov_file=None*, *dcmconfig=None*)
Perform actual conversion (calls to converter etc) given info from heuristic's *infotodict*

items symlink converter scaninfo_suffix custom_callable with_prov is_bids sourcedir outdir min_meta

None

heudiconv.convert.**convert_dicom**(*item_dicoms*, *bids_options*, *prefix*, *outdir*, *tempdirs*, *symlink*, *overwrite*)
Save DICOMs as output (default is by symbolic link)

item_dicoms [list of filenames] DICOMs to save

bids_options [list or None] If not None then save to BIDS format. List may be empty or contain bids specific options

prefix [string] Conversion outname

outdir [string] Output directory

tempdirs [TempDirs instance] Object to handle temporary directories created TODO: remove

symlink [bool] Create softlink to DICOMs - if False, create hardlink instead.

overwrite [bool] If True, allows overwriting of previous conversion

None

heudiconv.convert.**nipype_convert**(*item_dicoms*, *prefix*, *with_prov*, *bids_options*, *tmpdir*, *dcmconfig=None*)
Converts DICOMs grouped from heuristic using Nipype's Dcm2niix interface.

item_dicoms [List] DICOM files to convert

prefix [String] Heuristic output path

with_prov [Bool] Store provenance information

bids_options [List or None] If not None then output BIDS sidecar JSONs List may contain bids specific options

tmpdir [Directory] Conversion working directory

dcmconfig [File (optional)] JSON file used for additional Dcm2niix configuration

heudiconv.convert.**save_converted_files**(*res*, *item_dicoms*, *bids_options*, *outtype*, *prefix*, *outname_bids*, *overwrite*)

Copy converted files from tempdir to output directory. Will rename files if necessary.

res [Node] Nipype conversion Node with results

item_dicoms: list of filenames DICOMs converted

bids [list or None] If not list save to BIDS List may contain bids specific options

prefix : string

bids_outfiles Converted BIDS files

heudiconv.convert.**update_complex_name**(*metadata*, *filename*, *suffix*)

Insert *_part-<mag|phase>* entity into filename if data are from a sequence with magnitude/phase part.

metadata [dict] Scan metadata dictionary from BIDS sidecar file.

filename [str] Incoming filename

suffix [str] An index used for cases where a single scan produces multiple files, but the differences between those files are unknown.

filename [str] Updated filename with rec entity added in appropriate position.

heudiconv.convert.**update_multiecho_name**(*metadata*, *filename*, *echo_times*)

Insert *_echo-<num>* entity into filename if data are from a multi-echo sequence.

metadata [dict] Scan metadata dictionary from BIDS sidecar file.

filename [str] Incoming filename

echo_times [list] List of all echo times from scan. Used to determine the echo *number* (i.e., index) if field is missing from metadata.

filename [str] Updated filename with echo entity added, if appropriate.

heudiconv.convert.**update_uncombined_name**(*metadata*, *filename*, *channel_names*)

Insert *_ch-<num>* entity into filename if data are from a sequence with "save uncombined".

metadata [dict] Scan metadata dictionary from BIDS sidecar file.

filename [str] Incoming filename

channel_names [list] List of all channel names from scan. Used to determine the channel *number* (i.e., index) if field is missing from metadata.

filename [str] Updated filename with ch entity added, if appropriate.

3.6.3 DICOMS

heudiconv.dicoms.**compress_dicoms**(*dicom_list*, *out_prefix*, *tempdirs*, *overwrite*)

Archives DICOMs into a tarball

Also tries to do it reproducibly, so takes the date for files and target tarball based on the series time (within the first file)

dicom_list [list of str] list of dicom files

out_prefix [str] output path prefix, including the portion of the output file name before .dicom.tgz suffix

tempdirs [object] TempDirs object to handle multiple tmpdirs

overwrite [bool] Overwrite existing tarfiles

filename [str] Result tarball

heudiconv.dicoms.**create_seqinfo**(*mw*, *series_files*, *series_id*)
Generate sequence info

mw: MosaicWrapper series_files: list series_id: str

heudiconv.dicoms.**embed_dicom_and_nifti_metadata**(*dcmfiles*, *niftifile*, *infofile*, *bids_info*)
Embed metadata from nifti (affine etc) and dicoms into infofile (json)

niftifile should exist. Its affine's orientation information is used while establishing new *NiftiImage* out of dicom stack and together with *bids_info* (if provided) is dumped into json *infofile*

dcmfiles niftifile infofile bids_info: dict

Additional metadata to be embedded. *infofile* is overwritten if exists, so here you could pass some metadata which would overload (at the first level of the dict structure, no recursive fancy updates) what is obtained from nifti and dicoms

heudiconv.dicoms.**embed_metadata_from_dicoms**(*bids_options*, *item_dicoms*, *outname*, *outname_bids*, *prov_file*, *scaninfo*, *tempdirs*, *with_prov*)
Enhance sidecar information file with more information from DICOMs

bids_options item_dicoms outname outname_bids prov_file scaninfo tempdirs with_prov

heudiconv.dicoms.**get_dicom_series_time**(*dicom_list*)
Get time in seconds since epoch from dicom series date and time Primarily to be used for reproducible time stamping

heudiconv.dicoms.**group_dicoms_into_seqinfos**(*files*, *grouping*, *file_filter=None*, *dcmfilter=None*, *flatten=False*, *custom_grouping=None*)
Process list of dicoms and return seqinfo and file group *seqinfo* contains per-sequence extract of fields from DICOMs which will be later provided into heuristics to decide on filenames

files [list of str] List of files to consider

grouping [{'studyUID', 'accession_number', 'all', 'custom'}] How to group DICOMs for conversion. If 'custom', see *custom_grouping* parameter.

file_filter [callable, optional] Applied to each item of filenames. Should return True if file needs to be kept, False otherwise.

dcmfilter [callable, optional] If called on dcm_data and returns True, it is used to set series_id

flatten [bool, optional] Creates a flattened *seqinfo* with corresponding DICOM files. True when invoked with *dicom_dir_template*.

custom_grouping: str or callable, optional grouping key defined within heuristic. Can be a string of a DICOM attribute, or a method that handles more complex groupings.

seqinfo [list of list] *seqinfo* is a list of info entries per each sequence (some entry there defines a key for *filegrp*)

filegrp [dict] *filegrp* is a dictionary with files groupped per each sequence

heudiconv.dicoms.**parse_private_csa_header**(*dcm_data*, *public_attr*, *private_attr*, *default=None*)
Parses CSA header in cases where value is not defined publicly

dcm_data [pydicom Dataset object] DICOM metadata

public_attr [string] non-private DICOM attribute

private_attr [string] private DICOM attribute

default (optional) default value if private_attr not found

val (default: empty string) private attribute value or default

heudiconv.dicoms.**validate_dicom**(*fl*, *dcmfilter*)
Parse DICOM attributes. Returns None if not valid.

3.6.4 Parsing

heudiconv.parser.**find_files**(*regex*, *topdir='.'*, *exclude=None*, *exclude_vcs=True*, *dirs=False*)
Generator to find files matching regex Parameters ———- regex: basestring exclude: basestring, optional

Matches to exclude

exclude_vcs: If True, excludes commonly known VCS subdirectories. If string, used as regex to exclude those files (regex: */.(?:git|gitattributes|svn|bzr|hg)(?:/|$)*)

topdir: basestring or list, optional Directory where to search

dirs: bool, optional Either to match directories as well as files

heudiconv.parser.**get_extracted_dicoms**(*fl*)
Given a list of files, possibly extract some from tarballs For 'classical' heudiconv, if multiple tarballs are provided, they correspond to different sessions, so here we would group into sessions and return pairs *sessionid*, *files* with *sessionid* being None if no "sessions" detected for that file or there was just a single tarball in the list

heudiconv.parser.**get_study_sessions**(*dicom_dir_template*, *files_opt*, *heuristic*, *outdir*, *session*, *sids*, *grouping='studyUID'*)
Given options from cmdline sort files or dicom seqinfos into study_sessions which put together files for a single session of a subject in a study Two major possible workflows: - if dicom_dir_template provided – doesn't pre-load DICOMs and just

loads files pointed by each subject and possibly sessions as corresponding to different tarballs

- if files_opt is provided, sorts all DICOMs it can find under those paths

3.6.5 Batch Queuing

heudiconv.queue.**clean_args**(*hargs*, *iterarg*, *iteridx*)
Filters arguments for batch submission.

hargs: list Command-line arguments

iterarg: str Multi-argument to index (*subjects* OR *files*)

iteridx: int *iterarg* index to submit

cmdargs [list] Filtered arguments for batch submission

```
>>> from heudiconv.queue import clean_args
>>> cmd = ['heudiconv', '-d', '/some/{subject}/path',
...                     '-q', 'SLURM',
...                     '-s', 'sub-1', 'sub-2', 'sub-3', 'sub-4']
>>> clean_args(cmd, 'subjects', 0)
['heudiconv', '-d', '/some/{subject}/path', '-s', 'sub-1']
```

heudiconv.queue.**queue_conversion**(*queue*, *iterarg*, *iterables*, *queue_args=None*)

Write out conversion arguments to file and submit to a job scheduler. Parses *sys.argv* for heudiconv arguments.

queue: string Batch scheduler to use

iterarg: str Multi-argument to index (*subjects* OR *files*)

iterables: int Number of *iterarg* arguments

queue_args: string (optional) Additional queue arguments for job submission

3.6.6 Utility

Utility objects and functions

class heudiconv.utils.**File**(*name*, *executable=False*)

Helper for a file entry in the create_tree/@with_tree

It allows to define additional settings for entries

class heudiconv.utils.**SeqInfo**(*total_files_till_now*, *example_dcm_file*, *series_id*, *dcm_dir_name*, *series_files*, *unspecified*, *dim1*, *dim2*, *dim3*, *dim4*, *TR*, *TE*, *protocol_name*, *is_motion_corrected*, *is_derived*, *patient_id*, *study_description*, *referring_physician_name*, *series_description*, *sequence_name*, *image_type*, *accession_number*, *patient_age*, *patient_sex*, *date*, *series_uid*, *time*)

TE
Alias for field number 11

TR
Alias for field number 10

accession_number
Alias for field number 21

date
Alias for field number 24

dcm_dir_name
Alias for field number 3

dim1
Alias for field number 6

dim2
Alias for field number 7

dim3
Alias for field number 8

dim4
Alias for field number 9

example_dcm_file
Alias for field number 1

image_type
Alias for field number 20

is_derived
Alias for field number 14

is_motion_corrected
Alias for field number 13

patient_age
Alias for field number 22

patient_id
Alias for field number 15

patient_sex
Alias for field number 23

protocol_name
Alias for field number 12

referring_physician_name
Alias for field number 17

sequence_name
Alias for field number 19

series_description
Alias for field number 18

series_files
Alias for field number 4

series_id
Alias for field number 2

series_uid
Alias for field number 25

study_description
Alias for field number 16

time
Alias for field number 26

total_files_till_now
Alias for field number 0

unspecified
Alias for field number 5

class heudiconv.utils.**StudySessionInfo**(*locator*, *session*, *subject*)

locator
Alias for field number 0

session
Alias for field number 1

subject
Alias for field number 2

class heudiconv.utils.**TempDirs**
A helper to centralize handling and cleanup of dirs

heudiconv.utils.**assure_no_file_exists**(*path*)
Check if file or symlink (git-annex?) exists, and if so – remove

heudiconv.utils.**clear_temp_dicoms**(*item_dicoms*)
Ensures DICOM temporary directories are safely cleared

heudiconv.utils.**create_file_if_missing**(*filename*, *content*)
Create file if missing, so we do not override any possibly introduced changes

heudiconv.utils.**create_tree**(*path*, *tree*, *archives_leading_dir=True*)
Given a list of tuples (name, load) or a dict create such a tree

if load is a tuple or a dict itself – that would create either a subtree or an archive with that content and place it into the tree if name ends with .tar.gz

heudiconv.utils.**docstring_parameter**(**sub*)
Borrowed from https://stackoverflow.com/a/10308363/6145776

heudiconv.utils.**get_datetime**(*date*, *time*, ***, *microseconds=True*)
Combine date and time from dicom to isoformat.

date [str] Date in YYYYMMDD format.

time [str] Time in either HHMMSS.ffffff format or HHMMSS format.

microseconds: bool, optional Either to include microseconds in the output

datetime_str [str] Combined date and time in ISO format, with microseconds as if fraction was provided in 'time', and 'microseconds' was True.

heudiconv.utils.**get_known_heuristic_names**()
Return a list of heuristic names present under heudiconv/heuristics

heudiconv.utils.**get_typed_attr**(*obj*, *attr*, *_type*, *default=None*)
Typecasts an object's named attribute. If the attribute cannot be converted, the default value is returned instead.

obj: Object attr: Attribute _type: Type default: value, optional

heudiconv.utils.**is_readonly**(*path*)
Return True if it is a fully read-only file (dereferences the symlink)

heudiconv.utils.**json_dumps**(*json_obj*, *indent=2*, *sort_keys=True*)
Unified (default indent and sort_keys) invocation of json.dumps

heudiconv.utils.**json_dumps_pretty**(*j*, *indent=2*, *sort_keys=True*)
Given a json structure, pretty print it by colliding numeric arrays into a line.

If resultant structure differs from original – throws exception

heudiconv.utils.**load_heuristic**(*heuristic*)
Load heuristic from the file, return the module

heudiconv.utils.**load_json**(*filename*, *retry=0*)
Load data from a json file

filename [str] Filename to load data from.

retry: int, optional Number of times to retry opening/loading the file in case of failure. Code will sleep for 0.1 seconds between retries. Could be used in code which is not sensitive to order effects (e.g. like populating bids templates where the last one to do it, would make sure it would be the correct/final state).

data : dict

heudiconv.utils.**safe_copyfile**(*src*, *dest*, *overwrite=False*)
Copy file but blow if destination name already exists

heudiconv.utils.**safe_movefile**(*src*, *dest*, *overwrite=False*)
Move file but blow if destination name already exists

heudiconv.utils.**save_json**(*filename*, *data*, *indent=2*, *sort_keys=True*, *pretty=False*)
Save data to a json file

filename [str] Filename to save data in.

data [dict] Dictionary to save in json file.

indent : int, optional sort_keys : bool, optional pretty : bool, optional

heudiconv.utils.**set_readonly**(*path*, *read_only=True*)
Make file read only or writeable while preserving "access levels"

So if file was not readable by others, it should remain not readable by others.

path : str read_only : bool, optional

If True (default) - would make it read-only. If False, would make it writeable for levels where it is readable

heudiconv.utils.**slim_down_info**(*j*)
Given an aggregated info structure, removes excessive details

Such as CSA fields, and SourceImageSequence which on Siemens files could be huge and not providing any additional immediately usable information. If needed, could be recovered from stored DICOMs

heudiconv.utils.**treat_infofile**(*filename*)
Tune up generated .json file (slim down, pretty-print for humans).

Python Module Index

h

Index