django$_{mri}$

Release 0.1.0

Jul 27, 2020

Contents:

CHAPTER 1

Overview

django_mri is a reusable Django app providing general-purpose MRI data management utilities.

The purpose of this app is to abstract away technical details and automate commonplace coversion and extraction functionalities required by the various neuroimaging analysis toolkits. In addition, it provides preconfigured analysis interfaces and pipelines with the help of django_analyses and nipype.

CHAPTER 2

Installation

1. Install from PyPi:

```
$ pip install django_mri
```

2. Add "django_mri" and "django_dicom" to your project's INSTALLED_APPS setting:

Listing 1: <project>/settings.py

```
INSTALLED_APPS = [
    ...,
    "django_dicom",
    "django_mri",
]
```

Note: *django_mri* uses django_dicom to manage data imported as DICOM files.

3. Include the app's URLconf in your prject *urls.py*:

Listing 2: <project>/urls.py

```
urlpatterns = [
    ...,
    path("api/", include("django_mri.urls", namespace="mri")),
]
```

4. Run:

```
$ python manage.py migrate
```

5. *[Optional]* Load preconfigured sequence types and analyses:

Listing 3: Django shell

```
>>> from django_mri.analysis.utils import load_mri_analyses
>>> from django_mri.models import SequenceType
>>> from django_mri.models.common_sequences import sequences

# Create analyses and pipelines
>>> load_mri_analyses()

# Create common MRI sequence definitions
>>> for sequence in sequences:
>>>     SequenceType.objects.create(**sequence)
```

6. Start the development server and visit http://127.0.0.1:8000/admin/.
7. Visit http://127.0.0.1:8000/api/mri/.

CHAPTER 3

Importing Data

Currently, *django_mri* supports importing raw data only as DICOM files. To import a directory of DICOM data, simply run:

Listing 1: Django shell

```
>>> from django_mri.models import Scan

>>> path = "/path/to/dicom/archive/"
>>> Scan.objects.import_path(path)
Importing DICOM data: <####>image [<##:##>, <##.##>image/s]

Successfully imported DICOM data from <path>!
Created:        <####>
```

CHAPTER 4

Reference

4.1 Subpackages

4.1.1 Analysis

Module contents

Preconfigured analyses and pipelines to be used with django_analyses.

Example

To import the analyses and pipelines to the database, start a Django shell session and run:

```
from django_mri.analysis.utils import load_mri_analyses

load_mri_analyses()
```

See also:

- django_analyses
- nipype

Subpackages

Interfaces

Module contents

Interface classes for various MRI data processing tools.

Subpackages

FSL

Module contents

Custom interface classes for FSL analyses.

Submodules

django_mri.analysis.interfaces.fsl.fast module

Definition of the *FastWrapper* class.

class django_mri.analysis.interfaces.fsl.fast.**FastWrapper**(***inputs*)

Bases: nipype.interfaces.fsl.preprocess.FAST

A simple subclass of nipype's FAST interface, tweaking the run() method's output slightly to make output specification easier.

run(**args*, ***kwargs*) → dict

Edits the returned results dictionary to simplify the output specification.

Returns FAST results

Return type dict

django_mri.analysis.interfaces.fsl.fsl_anat module

Definition of the *FslAnat* class.

class django_mri.analysis.interfaces.fsl.fsl_anat.**FslAnat**(*weak_bias: bool = False*, *no_reorient: bool = False*, *no_crop: bool = False*, *no_bias: bool = False*, *no_registration: bool = False*, *no_nonlinear_registration: bool = False*, *no_segmentation: bool = False*, *no_subcortical_segmentation: bool = False*, *no_search: bool = False*, *bias_field_smoothing: float = None*, *image_type: str = 'T1'*, *no_cleanup: bool = False*)

Bases: object

Custom interface class for the fsl_anat anatomical preprocessing script.

FLAGS = {'no_nonlinear_registration': 'nononlinreg', 'no_registration': 'noreg', 'no
Conversion dictionary between the verbose names given to the class initialization keyword arguments and the script's parameters.

FLAG_ATTRIBUTES = ('weak_bias', 'no_reorient', 'no_crop', 'no_bias', 'no_registration'
"Flags" indicate parameters that are specified without any arguments, i.e. they are a switch for some binary configuration.

FORCED_SUFFIX = '.anat'
A suffix given by *fsl_anat* and removed by the interface.

OUTPUT_FILES = {'bias_corrected_brain': 'T1_biascorr_brain.nii.gz', 'bias_corrected_b
A dictionary of the file names expected to be created by running the script.

fix_output_path(*destination: pathlib.Path*) → None
Removed the forced *.anat* suffix appended to the destination directory.

Parameters **destination** (*Path*) – Destination directory

generate_command(*scan*, *destination: pathlib.Path = None*) → str
Returns the command to be executed in order to run the analysis.

Parameters

- **scan** (NIfTI) – The scan to run the analysis on
- **destination** (*Path, optional*) – The directory in which output files should be created, by default None

Returns Command string

Return type str

generate_flags() → str
Returns a string containing the various flags configured for this instance.

Returns Execution command flag parameters

Return type str

generate_output_dict(*destination: pathlib.Path = None*) → dict
Returns a dictionary of the run's output files.

Parameters **destination** (*Path, optional*) – Destination directory, by default None

Returns Output files by key

Return type dict

run(*scan*, *destination: pathlib.Path = None*) → dict
Runs *fsl_anat* with the provided *scan* as input and *destination* as the destination directory.

Parameters

- **scan** (NIfTI) – Input scan
- **destination** (*Path, optional*) – Destination directory, by default None

Returns Output files by key

Return type dict

Raises RuntimeError – Run failure

django_mri.analysis.interfaces.fsl.topup module

Definition of the *TopupWrapper* class.

class django_mri.analysis.interfaces.fsl.topup.**TopupWrapper**(**args*, ***kwargs*)
Bases: nipype.interfaces.fsl.epi.TOPUP

A simple subclass of nipype's TOPUP interface, tweaking the interface's __init__() method to make input specification easier.

PHASE_ENCODING_DICT = {'i': 'x', 'j': 'y', 'k': 'z'}

fix_phase_encoding(*phase_encoding: str*) → str
Converts phase encoding values from *i, j, k* to *x, y, z*.

Parameters **phase_encoding** (*str*) – Phase encoding

Returns Coverted phase encoding

Return type str

MATLAB

Module contents

Interfaces for MATLAB functions.

Subpackages

SPM Interfaces

Module contents

Interfaces for SPM functions.

Subpackages

CAT12 Interfaces

Module contents

Interfaces for CAT12 functions.

See also:

- CAT12 manual

Subpackages

CAT12 Segmentation Interface

Module contents

Definition of an interface for the CAT12 segmentation function.

Subpackages

CAT12 Segmentation Interface Utilities

Module contents

Definition of an interface for the CAT12 segmentation function.

Submodules

django_mri.analysis.interfaces.matlab.spm.cat12.segmentation.utils.verbosify_output_dict module

A utility module created for the definition of the *verbosify_output_dict()* method.

django_mri.analysis.interfaces.matlab.spm.cat12.segmentation.utils.verbosify_output_dict.**ve**

Flattens and verbosifies the output files dictionary to facilitate integration with django_analyses.

Parameters **output_dict** (*dict*) – Output files by key

Returns Flat and verbose output files by key

Return type dict

Submodules

django_mri.analysis.interfaces.matlab.spm.cat12.segmentation.defaults module

Default values for the various keys required in the batch template.

django_mri.analysis.interfaces.matlab.spm.cat12.segmentation.messages module

A module storing strings used to display messages.

django_mri.analysis.interfaces.matlab.spm.cat12.segmentation.outputs module

Dictionaries containing CAT12 segmentation output file names by key.

django_mri.analysis.interfaces.matlab.spm.cat12.segmentation.outputs.**AUXILIARY_OUTPUT = {'**
Artifacts created during execution.

django_mri.analysis.interfaces.matlab.spm.cat12.segmentation.outputs.**SEGMENTATION_OUTPUT =**
Output file names by key.

django_mri.analysis.interfaces.matlab.spm.cat12.segmentation.segmentation module

Definition of the *Segmentation* class.

class django_mri.analysis.interfaces.matlab.spm.cat12.segmentation.segmentation.**Segmentatio**
Bases: django_mri.analysis.interfaces.matlab.spm.spm_procedure.SPMProcedure

An interface for the CAT12 segmentation function.

AUXILIARY_OUTPUT = {'batch_file': 'segmentation.m', 'reports': ['report/cat_{file_nam

BATCH_TEMPLATE_ID = 'CAT12 Segmentation'

DEFAULTS = {'accuracy': 'average', 'affine_preprocessing': 'rough', 'affine_regulari

DEFAULT_BATCH_FILE_NAME = 'segmentation.m'

OUTPUT_DEFINITIONS = {'cobra': ['label/catROI_{file_name}.mat', 'label/catROI_{file_na

REDUNDANT_LOG_PATTERN = 'catlog_main_*_log*.txt'

TRANSFORMATIONS = {'accuracy': {'average': 0.5, 'high': 0.75, 'ultra high': 1}, 'a

organize_output(*path: pathlib.Path*, *created_uncompressed_version: bool*, *destination: pathlib.Path*, *remove_redundant_logs: bool*, *verbose_output_dict: bool = False*) → dict
Organized output files after execution.

Parameters

- **path** (*Path*) – Input file path
- **created_uncompressed_version** (*bool*) – Whether an uncompressed version of the input value was created or not
- **destination** (*Path*) – Output files destination directory
- **remove_redundant_logs** (*bool*) – Whether to remove logs created during execution or not
- **verbose_output_dict** (*bool, optional*) – Whether to flatten the output dictionary to facilitate integration with django_analyses, by default False

Returns Output files by keys

Return type dict

remove_redundant_logs(*run_dir: pathlib.Path*)
Removed unnecessary logs created during execution.

Parameters **run_dir** (*Path*) – Output files destination

run(*path: pathlib.Path*, *destination: pathlib.Path = None*, *remove_redundant_logs: bool = True*, *verbose_output_dict: bool = False*) → dict
Run CAT12 segmentation on the provided *.nii* input file.

Parameters

- **path** (*Path*) – Input file
- **destination** (*Path, optional*) – Output file destination directory, by default None
- **remove_redundant_logs** (*bool, optional*) – Whether to remove some logs created during execution, by default True

- **verbose_output_dict** (*bool, optional*) – Whether to verbosify and flatten the output files dictionary, by default False

Returns Output files by key

Return type dict

transform_options() → dict
Apply any transformation defined in the *transformations* module to the options dictionary.

Returns Tranformed options dictionary

Return type dict

update_batch_template(*data_path: pathlib.Path*) → str
Returns a copy of the batch template, updated with the configured options and data path.

Parameters **data_path** (*Path*) – Path of the input *.nii* file

Returns Updated batch template

Return type str

validate_and_fix_input_data(*path: pathlib.Path*) → Tuple[pathlib.Path, bool]
Validate the provided data path and creates an unzipped version if required.

Parameters **path** (*Path*) – Path to input data

Returns Fixed input data path, whether is was unzipped or not

Return type Tuple[Path, bool]

django_mri.analysis.interfaces.matlab.spm.cat12.segmentation.transformations module

Definition of the *SEGMENTATION_TRANSFORMATIONS* dictionary.

django_mri.analysis.interfaces.matlab.spm.cat12.segmentation.transformations.**SEGMENTATION_T**
Transformation to apply to option values before editing the batch template.

CAT12 Utilities

Module contents

CAT12 segmentation interface utilities.

Submodules

django_mri.analysis.interfaces.matlab.spm.cat12.utils.batch_templates module

Definition of the *CAT12_TEMPLATES* constant, used to locate batch template file paths by interface key.

django_mri.analysis.interfaces.matlab.spm.cat12.utils.batch_templates.**CAT12_TEMPLATES = {'C**
Batch template file path by interface (string) key.

django_mri.analysis.interfaces.matlab.spm.cat12.utils.template_files module

Relative paths to files required by some CAT12 analysis configurations.

`django_mri.analysis.interfaces.matlab.spm.cat12.utils.template_files.`**`RELATIVE_DARTEL_TEMPLA`**
Relative path to dartel template.

`django_mri.analysis.interfaces.matlab.spm.cat12.utils.template_files.`**`RELATIVE_TISSUE_PROBAI`**
Relative path to tissue probability map.

SPM Utilities

Module contents

Utilities for the *spm* module.

Submodules

django_mri.analysis.interfaces.matlab.spm.utils.batch_templates module

Definition of the *TEMPLATES* constant.

`django_mri.analysis.interfaces.matlab.spm.utils.batch_templates.`**`TEMPLATES = {'CAT12 Segment`**
Batch templates dictionary, associating interface keys with template paths.

django_mri.analysis.interfaces.matlab.spm.utils.nifti_validator module

Definition of the *NiftiValidator* class.

class `django_mri.analysis.interfaces.matlab.spm.utils.nifti_validator.`**`NiftiValidator`**(*allow_g*
bool
=
False)

Bases: `object`

A utility class used to validate *.nii* inputs and uncompress them if necessary.

`validate_and_fix`(*path: pathlib.Path*) → pathlib.Path
Validates the provided path represents a *.nii* file and uncompresses it if necessary.

Parameters **`path`** (*Path*) – *.nii* file path

Returns *.nii* file path

Return type Path

Raises

- `ValueError` – Invalid suffix (not *.nii*)
- `FileNotFoundError` – File does not exist

`validate_extension`(*path: pathlib.Path*) → bool
Validate the given file's extension.

Parameters **`path`** (*Path*) – File path to check

Returns Valid or invalid

Return type bool

MRtrix3

Module contents

Interfaces for MRtrix3 scripts.

See also:

- MRtrix3 user documentation

Submodules

django_mri.analysis.interfaces.mrtrix3.dwifslpreproc module

Definition of the *dwifslpreproc* interface.

class django_mri.analysis.interfaces.mrtrix3.dwifslpreproc.**DwiFslPreproc**(*configuration: dict*)

Bases: object

An interface for the MRtrix3 *dwifslpreproc* script.

References

- dwifslpreproc

DEFAULT_OUTPUT_NAME = 'preprocessed_dwi.mif'
Default name for primary output file.

EDDY_OUTPUTS = {'eddy_mask': 'eddy_mask.nii', 'out_movement_rms': 'eddy_movement_rms
eddy output files by key.

References

- eddy

FLAGS = ('align_seepi', 'rpe_none', 'rpe_pair', 'rpe_all', 'rpe_header', 'force', 'qui
"Flags" indicate parameters that are specified without any arguments, i.e. they are a switch for some binary configuration.

SUPPLEMENTARY_OUTPUTS = ('eddyqc_text', 'eddyqc_all')
Non-default output configurations.

add_supplementary_outputs(*destination: pathlib.Path*) → dict
Adds the *eddy* output files to the interface's configuration before generating the command to run.

References

- eddy

Parameters **destination** (*Path*) – Output directory

Returns Updated configuration dictionary

Return type dict

generate_command(*scan*, *destination: pathlib.Path*, *config: str*) → str
Returns the command to be executed in order to run the analysis.

Parameters

- **scan** (Scan) – Input scan
- **destination** (*Path*) – Output files destination direcotry
- **config** (*str*) – Configuration arguments for the command

Returns Complete execution command

Return type str

generate_output_dict(*destination: pathlib.Path*) → dict
Generates a dictionary of the expected output file paths by key.

Parameters **destination** (*Path*) – Output files destination directory

Returns Output files by key

Return type dict

run(*scan*, *destination: pathlib.Path = None*) → dict
Runs *dwifslpreproc* with the provided *scan* as input. If *destination* is not specified, output files will be created within *scan*'s directory.

Parameters

- **scan** (Scan) – Input scan
- **destination** (*Path, optional*) – Output files destination directory, by default None

Returns Output files by key

Return type dict

Raises RuntimeError – Run failure

Submodules

django_mri.analysis.interfaces.dcm2niix module

Definition of the *Dcm2niix* class.

django_mri.analysis.interfaces.dcm2niix.**BASE_DIR = PosixPath('/home/docs/checkouts/readthe**
Project's base directory.

class django_mri.analysis.interfaces.dcm2niix.**Dcm2niix**(*path: pathlib.Path = PosixPath('/home/docs/checkouts/readthedocs.org/us mri/checkouts/latest/django_mri/utils/dcm2niix')*)

Bases: object

An interface for dcm2niix.

See also:

- dcm2niix user manual

BOOLEAN = {False: 'n', True: 'y'}
Convert boolean configurations to "y" or "n".

FLAGS = {'BIDS': '-b', 'compressed': '-z', 'directory': '-o', 'name': '-f'}
Arguemnts dictionary. Keys are the interface's verbose names and values are the actual CLI's arguments.

convert(*path: pathlib.Path*, *destination: pathlib.Path*, *compressed: bool = True*, *generate_json: bool = True*) → pathlib.Path
Coverts the series in the provided *path* from DICOM to NIfTI.

Parameters

- **path** (*Path*) – Input DICOM directory
- **destination** (*Path*) – Output destination directory
- **compressed** (*bool, optional*) – Whether to create compressed (*.nii.gz*) files or not, by default True
- **generate_json** (*bool, optional*) – Whether to generate a "BIDS sidecar" JSON file with supplementary information, by default True

Returns Output file path

Return type Path

Raises

- RuntimeError – *dcm2niix* run failure
- NotImplementedError – *dcm2niix* executable could not be found

extract_output_path(*stdout: str*, *compressed: bool*) → pathlib.Path
Returns the path of the output file.

Parameters

- **stdout** (*str*) – *dcm2niix* run output
- **compressed** (*bool*) – Whether the file is compressed (*.nii.gz*) or not, by default True

Returns Output file path

Return type Path

generate_command(*path: pathlib.Path*, *destination: pathlib.Path*, *compressed: bool = True*, *generate_json: bool = True*) → list
Generate the command to execute to run *dcm2niix*.

Parameters

- **path** (*Path*) – Input path
- **destination** (*Path*) – Output file destination
- **compressed** (*bool, optional*) – Whether to compress the file or not, by default True
- **generate_json** (*bool, optional*) – Whether to generate a JSON or not, by default True

Returns Command to run, split at spaces

Return type list

django_mri.analysis.interfaces.messages module

A module storing strings used to display messages.

Pipelines

Module contents

Proposed pipeline definitions for MRI data using commonplace neuroimaging analysis interfaces.

Pipeline definitions are aggregated in `pipeline_definitions` and may be added to the database using the `Pipeline.objects.from_list()` method.

Example

```
from django_analyses.models.pipeline import Pipeline
from django_mri.analysis.pipeline_definitions import pipeline_definitions

Pipeline.objects.from_list(pipeline_definitions)
```

Submodules

django_mri.analysis.pipelines.basic_fsl_preprocessing module

django_mri.analysis.pipelines.dwi_preprocessing module

Full DWI preprocessing pipeline.

Steps:

- Extract b0 from AP
- Merge AP_b0 and PA
- Create brain mask
- Convert to mif (merged, AP)
- Denoise initial AP
- *dwifslpreproc* AP merged
- Gibbs correction
- Bias correction

User inputs:

- *fslroi* node: in_file [AP]
- *fslmerge* node: in_files [PA]
- mrconvert_1 –> AP: in_bvec [bvec]
- mrconvert_1 –> AP: in_bval [bval]

django_mri.analysis.pipelines.fieldmap_correction module

Field-map correction for EPI scans using FSL.

Specifications

Module contents

Input and output specifications dictionary to be included in `analysis_definitions`.

References

- Input and Output Specification

See also:

- `Analysis.objects.from_dict()`
- `InputSpecification.objects.from_dict()`
- `OutputSpecification.objects.from_dict()`

Subpackages

FreeSurfer

Module contents

Input and output specification dicationaries for FreeSurfer analyses.

Submodules

django_mri.analysis.specifications.freesurfer.recon_all module

Input and output specification dictionaries for FreeSurfer's recon_all script.

`django_mri.analysis.specifications.freesurfer.recon_all.`**`RECON_ALL_INPUT_SPECIFICATION = {'I`**
recon_all input specification.

`django_mri.analysis.specifications.freesurfer.recon_all.`**`RECON_ALL_OUTPUT_SPECIFICATION = {`**
recon_all output specification.

FSL

Module contents

Input and output specification dicationaries for FSL analyses.

Submodules

django_mri.analysis.specifications.fsl.apply_topup module

Input and output specification dictionaries for FSL's *applytopup* script.

See also:

- nipype.interfaces.fsl.epi.ApplyTOPUP

Notes

For more information about *applytopup*, see FSL's TOPUP/ApplyTOPUP User Guide

`django_mri.analysis.specifications.fsl.apply_topup.`**`APPLY_TOPUP_INPUT_SPECIFICATION = {'data`**
applytopup input specification.

`django_mri.analysis.specifications.fsl.apply_topup.`**`APPLY_TOPUP_OUTPUT_SPECIFICATION = {'out`**
applytopup output specification.

django_mri.analysis.specifications.fsl.bet module

Input and output specification dictionaries for FSL's **BET_** script.

See also:

- nipype.interfaces.fsl.preprocess.BET

Notes

For more information about BET, see FSL's BET documentation.

`django_mri.analysis.specifications.fsl.bet.`**`BET_INPUT_SPECIFICATION = {'center':  {'descript`**
BET input specification dictionary.

`django_mri.analysis.specifications.fsl.bet.`**`BET_OUTPUT_SPECIFICATION = {'inskull_mask_file'`**
BET output specification dictionary.

django_mri.analysis.specifications.fsl.binary_maths module

Input and output specification dictionaries for nipype's BinaryMaths interface, wrapping FSL's fslmaths.

`django_mri.analysis.specifications.fsl.binary_maths.`**`BINARY_MATHS_INPUT_SPECIFICATION = {'in`**
BinaryMaths input specification dictionary.

`django_mri.analysis.specifications.fsl.binary_maths.`**`BINARY_MATHS_OUTPUT_SPECIFICATION = {'o`**
BinaryMaths input specification dictionary.

django_mri.analysis.specifications.fsl.eddy module

Input and output specification dictionaries for FSL's eddy script.

`django_mri.analysis.specifications.fsl.eddy.`**`EDDY_INPUT_SPECIFICATION = {'cnr_maps':  {'defa`**
eddy input specification dictionary.

django_mri.analysis.specifications.fsl.eddy.**EDDY_OUTPUT_SPECIFICATION = {'out_cnr_maps':**
eddy input specification dictionary.

django_mri.analysis.specifications.fsl.fast module

Input and output specification dictionaries for FSL's FAST script.

django_mri.analysis.specifications.fsl.fast.**FAST_INPUT_SPECIFICATION = {'args': {'descript**
FAST input specification dictionary.

django_mri.analysis.specifications.fsl.fast.**FAST_OUTPUT_SPECIFICATION = {'mixeltype': {'de**
FAST output specification dictionary.

django_mri.analysis.specifications.fsl.flirt module

Input and output specification dictionaries for FSL's FLIRT script.

django_mri.analysis.specifications.fsl.flirt.**FLIRT_INPUT_SPECIFICATION = {'angle_rep': {'**
FLIRT input specification dictionary.

django_mri.analysis.specifications.fsl.flirt.**FLIRT_OUTPUT_SPECIFICATION = {'out_file': {'**
FLIRT output specification dictionary.

django_mri.analysis.specifications.fsl.fnirt module

Input and output specification dictionaries for FSL's FNIRT script.

django_mri.analysis.specifications.fsl.fnirt.**FNIRT_INPUT_SPECIFICATION = {'affine_file':**
FNIRT input specification dictionary.

django_mri.analysis.specifications.fsl.fnirt.**FNIRT_OUTPUT_SPECIFICATION = {'field_file':**
FNIRT output specification dictionary.

django_mri.analysis.specifications.fsl.fsl_anat module

Input and output specification dictionaries for FSL's fsl_anat script.

django_mri.analysis.specifications.fsl.fsl_anat.**FSL_ANAT_INPUT_SPECIFICATION = {'bias_field**
fsl_anat input specification dictionary.

django_mri.analysis.specifications.fsl.fsl_anat.**FSL_ANAT_OUTPUT_SPECIFICATION = {'bias_cor**
fsl_anat output specification dictionary.

django_mri.analysis.specifications.fsl.fslmerge module

Input and output specification dictionaries for FSL's fslmerge script.

django_mri.analysis.specifications.fsl.fslmerge.**FSLMERGE_INPUT_SPECIFICATION = {'dimension**
fslmerge input specification dictionary.

django_mri.analysis.specifications.fsl.fslmerge.**FSLMERGE_OUTPUT_SPECIFICATION = {'merged_fi**
fslmerge output specification dictionary.

django_mri.analysis.specifications.fsl.fslroi module

Input and output specification dictionaries for FSL's fslroi script.

`django_mri.analysis.specifications.fsl.fslroi.`**`FSLROI_INPUT_SPECIFICATION = {'crop_list':`**
fslroi input specification dictionary.

`django_mri.analysis.specifications.fsl.fslroi.`**`FSLROI_OUTPUT_SPECIFICATION = {'roi_file':`**
fslroi output specification dictionary.

django_mri.analysis.specifications.fsl.mean_image module

Input and output specification dictionaries for nipype's MeanImage interface, wrapping FSL's fslmaths.

`django_mri.analysis.specifications.fsl.mean_image.`**`MEAN_IMAGE_INPUT_SPECIFICATION = {'dimens`**
MeanImage input specification dictionary.

`django_mri.analysis.specifications.fsl.mean_image.`**`MEAN_IMAGE_OUTPUT_SPECIFICATION = {'out_`**
MeanImage output specification dictionary.

django_mri.analysis.specifications.fsl.reorient2std module

Input and output specification dictionaries for nipype's Reorient2Std interface, wrapping FSL's fslreorient2std.

`django_mri.analysis.specifications.fsl.reorient2std.`**`REORIENT2STD_INPUT_SPECIFICATION = {'a`**
Reorient2Std input specification dictionary.

`django_mri.analysis.specifications.fsl.reorient2std.`**`REORIENT2STD_OUTPUT_SPECIFICATION = {'`**
Reorient2Std output specification dictionary.

django_mri.analysis.specifications.fsl.robustfov module

Input and output specification dictionaries for nipype's RobustFOV interface, wrapping FSL's *robustfov*.

`django_mri.analysis.specifications.fsl.robustfov.`**`ROBUSTFOV_INPUT_SPECIFICATION = {'args':`**
RobustFOV input specification dictionary.

`django_mri.analysis.specifications.fsl.robustfov.`**`ROBUSTFOV_OUTPUT_SPECIFICATION = {'out_ro`**
RobustFOV output specification dictionary.

django_mri.analysis.specifications.fsl.susan module

Input and output specification dictionaries for FSL's SUSAN script.

`django_mri.analysis.specifications.fsl.susan.`**`SUSAN_INPUT_SPECIFICATION = {'args': {'descr`**
SUSAN input specification dictionary.

`django_mri.analysis.specifications.fsl.susan.`**`SUSAN_OUTPUT_SPECIFICATION = {'smoothed_file'`**
SUSAN output specification dictionary.

django_mri.analysis.specifications.fsl.topup module

MRtrix3

Module contents

Input and output specification dicationaries for MRtrix3 analyses.

Submodules

django_mri.analysis.specifications.mrtrix3.bias_correct module

Input and output specification dictionaries for MRtrix's *dwibiascorrect* script.

See also:

- nipype.interfaces.mrtrix3.preprocess.DWIBiasCorrect

Notes

For more information, see MRtrix3's dwibiascorrect reference.

`django_mri.analysis.specifications.mrtrix3.bias_correct.`**`BIAS_CORRECT_INPUT_SPECIFICATION =`**
DWIBiasCorrect input specification dictionary.

`django_mri.analysis.specifications.mrtrix3.bias_correct.`**`BIAS_CORRECT_OUTPUT_SPECIFICATION =`**
DWIBiasCorrect output specification dictionary.

django_mri.analysis.specifications.mrtrix.degibbs module

Input and output specification dictionaries for MRtrix's *mrdegibbs* script.

See also:

- nipype.interfaces.mrtrix3.preprocess.MRDeGibbs

Notes

For more information, see MRtrix3's mrdegibbs reference.

`django_mri.analysis.specifications.mrtrix3.degibbs.`**`DEGIBBS_INPUT_SPECIFICATION = {'axes':`**
MRDeGibbs input specification.

`django_mri.analysis.specifications.mrtrix3.degibbs.`**`DEGIBBS_OUTPUT_SPECIFICATION = {'out_fi`**
MRDeGibbs output specification.

django_mri.analysis.specifications.mrtrix.denoise module

Input and output specification dictionaries for MRtrix's *dwidenoise* script.

See also:

- nipype.interfaces.mrtrix3.preprocess.DWIDenoise

Notes

For more information, see MRtrix3's dwidenoise reference.

django_mri.analysis.specifications.mrtrix.dwifslpreproc module

Input and output specification dictionaries for MRtrix's *dwifslpreproc* script.

See also:

- *DwiFslPreproc*

Notes

For more information, see MRtrix3's dwifslpreproc reference.

`django_mri.analysis.specifications.mrtrix3.dwifslpreproc.`**`DWIFSLPREPROC_INPUT_SPECIFICATION`**
DwiFslPreproc input specification dictionary.

`django_mri.analysis.specifications.mrtrix3.dwifslpreproc.`**`DWIFSLPREPROC_OUTPUT_SPECIFICATIO`**
DwiFslPreproc output specification dictionary.

django_mri.analysis.specifications.mrtrix.mrconvert module

Input and output specification dictionaries for MRtrix's *mrconvert* script.

See also:

- nipype.interfaces.mrtrix3.utils.MRConvert

Notes

For more information, see MRtrix3's mrconvert reference.

`django_mri.analysis.specifications.mrtrix3.mrconvert.`**`MRCONVERT_INPUT_SPECIFICATION = {'axes`**
MRConvert input specification dictionary.

`django_mri.analysis.specifications.mrtrix3.mrconvert.`**`MRCONVERT_OUTPUT_SPECIFICATION = {'out`**
MRConvert output specification dictionary.

SPM

Module contents

Input and output specifications for SPM functions.

Subpackages

CAT12

Module contents

Input and output specifications for CAT12 functions.

Submodules

django_mri.analysis.specifications.spm.cat12.segmentation module

Input and output specification dictionaries for CAT12 segmentation interface.

See also:

- *CAT12 segmentation interface*

`django_mri.analysis.specifications.spm.cat12.segmentation.`**`CAT12_SEGMENTATION_INPUT_SPECIFIC`**
CAT12 segmentation interface input specification dictionary.

`django_mri.analysis.specifications.spm.cat12.segmentation.`**`CAT12_SEGMENTATION_OUTPUT_SPECIFI`**
CAT12 segmentation interface output specification dictionary.

Utilities

Module contents

Submodules

django_mri.analysis.utils.get_latest_analysis_version module

Definition of the `get_lastest_analysis_version()` function.

`django_mri.analysis.utils.get_latest_analysis_version.`**`get_lastest_analysis_version`**(*analysis: Union[str, int, django_ana* → django_ana

Returns the "lastest" (first by descending title order) analysis version of the provided analysis.

Analysis may be specified providing either its primary key, title, or an actual `Analysis` instance.

Parameters **`analysis`** (*`Union[str, int, Analysis]`*) – The desired analysis

Returns First analysis version

Return type AnalysisVersion

Raises `ValueError` – No analysis version found

django_mri.analysis.utils.get_mrconvert_node

django_mri.analysis.utils.load_mri_analyses

Submodules

django_mri.analysis.analysis_definitions module

django_mri.analysis.mri_interfaces module

Each analysis version imported to the database using django_analyses needs to have a matching interface registered for it in the project's settings. This interface is expected to be some class exposing a method (by default `run()`) which returns a dictionary of outputs matching its associated `OutputSpecification`.

References

- Simplified Analysis Integration Example

`django_mri.analysis.mri_interfaces.`**`interfaces = {'BET': {None:  <class 'nipype.interfaces.`**

A dictionary that should be imported in the project's settings and included within the *ANALYSIS_INTERFACES* setting.

django_mri.analysis.messages module

A module storing strings used to display messages.

django_mri.analysis.pipeline_definitions module

django_mri.analysis.visualizers module

class `django_mri.analysis.visualizers.`**`FslAnatVisualizer`**(*run: django_analyses.models.run.Run*)

Bases: `object`

`input_image`

`output_images`

`visualize``()` → None

4.1.2 Filters

Module contents

Filters for app's models.

Notes

For more information, see:

- Django REST Framework filtering documentation.
- django-filter's documentation for Integration with DRF.

Submodules

django_mri.filters.scan_filter module

Definition of the *ScanFilter* class.

class django_mri.filters.scan_filter.**NumberInFilter**(**args*, ***kwargs*)
Bases: django_filters.filters.BaseInFilter, django_filters.filters.NumberFilter

class django_mri.filters.scan_filter.**ScanFilter**(*data=None*, *queryset=None*, ***, *request=None*, *prefix=None*)
Bases: django_filters.rest_framework.filterset.FilterSet

Provides useful filtering options for the Series class.

base_filters = {'created': <django_filters.filters.DateTimeFromToRangeFilter object>,

declared_filters = {'created': <django_filters.filters.DateTimeFromToRangeFilter obje

django_mri.filters.scan_filter.**filter_by_sequence_type**(*queryset*, *field_name*, *value*)

django_mri.filters.sequence_type_filter module

4.1.3 Models

Module contents

Definition of the app's models.

Subpackages

Choices

Module contents

Choice ENUMs for easier maintenance of CharField's choice parameters.

Submodules

django_mri.models.choices.scanning_sequence module

A ChoiceEnum to represent ScanningSequence values.

class django_mri.models.choices.scanning_sequence.**ScanningSequence**
Bases: dicom_parser.utils.choice_enum.ChoiceEnum

An enumeration.

EP = 'Echo Planar'

GR = 'Gradient Recalled'

IR = 'Inversion Recovery'

RM = 'Research Mode'

SE = 'Spin Echo'

django_mri.models.choices.sequence_variant module

A ChoiceEnum to represent SequenceVariant values.

class django_mri.models.choices.sequence_variant.**SequenceVariant**
Bases: dicom_parser.utils.choice_enum.ChoiceEnum

An enumeration.

MP = 'MAG Prepared'

MTC = 'Magnetization Transfer Contrast'

NONE = 'None'

OSP = 'Oversampling Phase'

SK = 'Segmented k-Space'

SP = 'Spoiled'

SS = 'Steady State'

TRSS = 'Time Reversed Steady State'

Inputs

Module contents

Custom Input and InputDefinition subclasses.

These models expand upon django_analyses' input module to facilitate integration with the various analysis interfaces.

Submodules

django_mri.models.inputs.nifti_input module

class django_mri.models.inputs.nifti_input.**NiftiInput**(*id*, *run*, *input_ptr*, *value*, *definition*)
Bases: django_analyses.models.input.input.Input

definition
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

value
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

django_mri.models.inputs.nifti_input_definition module

class django_mri.models.inputs.nifti_input_definition.**NiftiInputDefinition**(*id*, *key*, *required*, *description*, *is_configuration*, *value_attribute*, *db_value_preprocessing*, *run_method_input*, *input_definition_ptr*)

Bases: `django_analyses.models.input.definitions.input_definition.InputDefinition`

input_class
alias of *django_mri.models.inputs.nifti_input.NiftiInput*

input_set
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

django_mri.models.inputs.scan_input module

class django_mri.models.inputs.scan_input.**ScanInput**(*id*, *run*, *input_ptr*, *value*, *definition*)

Bases: `django_analyses.models.input.input.Input`

definition
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

value
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

django_mri.models.inputs.scan_input_definition module

class django_mri.models.inputs.scan_input_definition.**ScanInputDefinition**(*id*, *key*, *required*, *description*, *is_configuration*, *value_attribute*, *db_value_preprocessing*, *run_method_input*, *input_definition_ptr*)

Bases: `django_analyses.models.input.definitions.input_definition.InputDefinition`

input_class
alias of *django_mri.models.inputs.scan_input.ScanInput*

input_set
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

Managers

Module contents

Submodules

django_mri.models.managers.scan module

class django_mri.models.managers.scan.**ScanManager**

Bases: `django.db.models.manager.Manager`

import_dicom_data(*path: pathlib.Path*, *progressbar: bool = True*, *report: bool = True*) → tuple

import_path(*path: pathlib.Path*, *progressbar: bool = True*, *report: bool = True*) → tuple

Outputs

Module contents

Custom `Output` and `OutputDefinition` subclasses.

These models expand upon django_analyses' `output` module to facilitate integration with the various analysis interfaces.

Submodules

django_mri.models.outputs.nifti_output module

class django_mri.models.outputs.nifti_output.**NiftiOutput**(*id*, *run*, *output_ptr*, *value*, *definition*)

Bases: `django_analyses.models.output.output.Output`

definition

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

value

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

django_mri.models.outputs.nifti_output_definition module

class django_mri.models.outputs.nifti_output_definition.**NiftiOutputDefinition**(*id*, *key*, *description*, *output_definition_ptr*)

Bases: `django_analyses.models.output.definitions.output_definition.OutputDefinition`

output_class
alias of *django_mri.models.outputs.nifti_output.NiftiOutput*

output_set
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

pre_output_instance_create(*kwargs: dict*) → None

django_mri.models.outputs.output_definitions module

class django_mri.models.outputs.output_definitions.**OutputDefinitions**

Bases: `enum.Enum`

An enumeration.

NIFTI = 'NIfTI'

SCAN = 'Scan'

Submodules

django_mri.models.common_sequences module

A list of common MRI sequences and their respective DICOM header attributes.

django_mri.models.fields module

Definition of a custom `ArrayField` subclass.

Copied from https://blogs.gnome.org/danni/2016/03/08/multiple-choice-using-djangos-postgres-arrayfield/.

class django_mri.models.fields.**ChoiceArrayField**(*base_field*, *size=None*, ***kwargs*)
Bases: `django.contrib.postgres.fields.array.ArrayField`

A field that allows us to store an array of choices.

Uses Django's PostgreSQL ArrayField and a MultipleChoiceField for its formfield.

formfield(***kwargs*)
Return a django.forms.Field instance for this field.

django_mri.models.help_text module

A module storing strings used to populate the fields' help_text attributes.

django_mri.models.messages module

A module storing strings used to display messages.

django_mri.models.nifti module

Definition of the *NIfTI* model.

class django_mri.models.nifti.**NIfTI**(**args*, ***kwargs*)
Bases: `django_extensions.db.models.TimeStampedModel`

A model representing a NIfTI file in the database.

b_value
Returns the B-value of DWI scans as calculated by dcm2niix.

See also:

- *get_b_value()*

Returns B-value

Return type List[int]

b_vector
Returns the B-vector of DWI scans as calculated by dcm2niix.

See also:

- *get_b_vector()*

Returns B-vector

Return type List[List[float]]

compress(*keep_source: bool = False*) → pathlib.Path
Compress the associated *.nii* using gzip, if it isn't already compressed.

Parameters **keep_source** (*bool, optional*) – Whether to keep a copy of the uncompressed file, by default False

Returns Path of the compressed (*.nii.gz*) file

Return type Path

compressed
Compresses the associated *.nii* file using gzip if it isn't and returns its path.

Returns Compressed *.nii.gz* file associated with this instance

Return type Path

get_b_value() → List[int]
Returns the degree of diffusion weighting applied (b-value) for each diffusion direction. This method relies on dcm2niix's default configuration in which when diffusion-weighted images (DWI) are converted, another file with the same name and a "bval" extension is created alongside.

Hint: For more information, see dcm2niix's Diffusion Tensor Imaging section of the user guide.

See also:

- *b_value*

Returns b-value for each diffusion direction.

Return type List[int]

get_b_vector() → List[List[float]]
Returns the b-vectors representing the diffusion weighting gradient scheme. This method relies on dcm2niix's default configuration in which when diffusion-weighted images (DWI) are converted, another file with the same name and a "bvec" extension is created alongside.

Hint: For more information, see dcm2niix's Diffusion Tensor Imaging section of the user guide.

See also:

- *b_vector*

Returns b-value for each diffusion direction

Return type List[List[float]]

get_data() → numpy.ndarray
Uses NiBabel to return the underlying pixel data as a NumPy array.

Returns Pixel data.

Return type np.ndarray

get_effective_spacing() → float
Reads the effective echo spacing value extracted by dcm2niix upon conversion.

Returns Effective echo spacing

Return type float

get_phase_encoding_direction() → float
Reads the phase encoding direction value extracted by dcm2niix upon conversion.

Returns Phase encoding direction

Return type float

get_total_readout_time() → float
Reads the total readout time extracted by dcm2niix upon conversion.

Hint: Total readout time is defined as the time from the center of the first echo to the center of the last (in seconds).

Returns Total readout time

Return type float

is_compressed
Whether the associated *.nii* file is compressed with gzip or not.

Returns Associated *.nii* file gzip compression state

Return type bool

is_raw
Whether the created instance is the product of a direct conversion from some raw format to NIfTI or of a manipulation of the data.

json_data
Reads BIDS sidecar information and caches within a local variable to prevent multiple reads.

See also:

- *read_json()*

Returns "BIDS sidecar" JSON data

Return type dict

objects = <django.db.models.manager.Manager object>

path
Path of the *.nii* file within the application's media directory.

read_json() → dict
Returns the JSON data generated alognside *.nii* files generated using dcm2niix's *"BIDS sidecar"* option.

Notes

- For more information about dcm2niix and the BIDS sidecar, see dcm2niix's general usage manual.
- For more information about the extracted properties and their usage see Acquiring and Using Fieldmaps

Returns BIDS sidecar information stored in a JSON file, or *{}* if the file doesn't exist

Return type dict

run_input_set
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

run_output_set
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

scan
Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

uncompress(*keep_source: bool = False*) → pathlib.Path
Uncompress the associated *.nii* using gzip, if it isn't already uncompressed.

Parameters **keep_source** (*bool, optional*) – Whether to keep a copy of the compressed file, by default False

Returns Path of the uncompressed (*.nii*) file

Return type Path

uncompressed
Uncompresses the associated *.nii* file using gzip if it isn't and returns its path.

Returns Uncompressed *.nii* file associated with this instance

Return type Path

django_mri.models.scan module

Definition of the *Scan* model.

class django_mri.models.scan.**Scan**(**args*, ***kwargs*)
Bases: `django_extensions.db.models.TimeStampedModel`

A model used to represent an MRI scan independently from the file-format in which it is saved. This model handles any conversions between formats in case they are required, and allows for easy querying of MRI scans based on universal attributes.

added_by
Keeps a record of the user that added this scan.

comments
Any other comments about this scan.

compile_to_bids(*bids_path: pathlib.Path*)

convert_to_mif() → pathlib.Path
Creates a *.mif* version of this scan using mrconvert.

Returns Created file path

Return type Path

description
Short description of the scan's acquisition parameters.

dicom
If this instance's origin is a DICOM file, or it was saved as one, this field stores the association with the appropriate :class‘django_dicom.models.series.Series‘ instance.

dicom_to_nifti(*destination: pathlib.Path = None*, *compressed: bool = True*, *generate_json: bool = True*) → django_mri.models.nifti.NIfTI
Convert this scan from DICOM to NIfTI using _dcm2niix.

Parameters **destination** (*Path, optional*) – The desired path for conversion output (the default is None, which will create the file in some default location)

Raises AttributeError – If no DICOM series is related to this scan

Returns A `django_mri.NIfTI` instance referencing the conversion output

Return type *NIfTI*

echo_time
The time between the application of the radio-frequency excitation pulse and the peak of the signal induced in the coil (in milliseconds).

get_bids_destination() → pathlib.Path
Returns the BIDS-compatible destination of this scan's associated *NIfTI* file.

Returns BIDS-compatible NIfTI file destination

Return type pathlib.Path

get_default_mif_path() → pathlib.Path
Returns the default *.mif* path for this scan.

Returns Default *.mif* path

Return type Path

get_default_nifti_destination() → pathlib.Path
Returns the default path for a NIfTI version of this scan.

Returns Default path for NIfTI file

Return type str

get_default_nifti_dir() → pathlib.Path
Returns the default location for the creation of a NIfTI version of the scan. Currently only conversion from DICOM is supported.

Returns Default location for conversion output

Return type str

get_default_nifti_name() → str
Returns the default file name for a NIfTI version of this scan.

Returns Default file name

Return type str

infer_sequence_type() → django_mri.models.sequence_type.SequenceType
Tries to infer the sequence type using associated data.

Returns The inferred sequence type

Return type *SequenceType*

infer_sequence_type_from_dicom() → django_mri.models.sequence_type.SequenceType
Returns the appropriate `django_mri.SequenceType` instance according to the scan's "*ScanningSequence*" and "*SequenceVariant*" header values.

Returns The inferred sequence type

Return type *SequenceType*

institution_name
The institution in which this scan was acquired.

inversion_time
The time between the 180-degree inversion pulse and the following spin-echo (SE) sequence (in milliseconds).

is_updated_from_dicom
Keeps track of whether we've updated the instance's fields from DICOM header data or not.

mif
Returns the *.mif* version of this scan, creating it if it doesn't exist.

Returns *.mif* file path

Return type Path

See also:

- *convert_to_mif()*

nifti
Returns the associated *NIfTI* instance if one exists, or tries to create one if it doesn't.

Returns Associated NIfTI instance

Return type *NIfTI*

number
The relative number of this scan in the session in which it was acquired.

objects = <django_mri.models.managers.scan.ScanManager object>

repetition_time
The time between two successive RF pulses (in milliseconds).

run_input_set
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

save(**args*, ***kwargs*) → None
Overrides the model's `save()` method to provide custom validation.

Hint: For more information, see Django's documentation on overriding model methods.

---

sequence_type
Returns the sequence type instance fitting this scan if one exists.

See also:

- *infer_sequence_type()*
- *django_mri.models.sequence_type.SequenceType*

Returns Inferred sequence type

Return type *SequenceType*

spatial_resolution
The spatial resolution of the image in millimeters.

study_groups
Individual scans may be associated with multiple *Group* instances. This is meant to provide flexibility in managing access to data between researchers working on different studies. The *Group* model is expected to be specified as *STUDY_GROUP_MODEL* in the project's settings.

subject
Associates this scan with some subject. Subjects are expected to be represented by a model specified as *SUBJECT_MODEL* in the project's settings.

suggest_subject(*subject*) → None

time
Acquisition datetime.

update_fields_from_dicom() → None
Sets instance fields from related DICOM series.

Raises `AttributeError` – If not DICOM series is related to this scan

warn_subject_mismatch(*subject*)
Warns the user regarding a mismatch in subject identity.

Parameters **subject** (*django.db.models.Model*) – Suggested subject identity

django_mri.models.sequence_type module

class django_mri.models.sequence_type.**SequenceType**(**args*, ***kwargs*)
Bases: `django_extensions.db.models.TitleDescriptionModel`, `django_extensions.db.models.TimeStampedModel`

The purpose of this model is to provide a title and description for commonly used sequences, as well as to facilitate queries.

Each particular sequence is defined as a unique combination of DICOM's "*ScanningSequence*" and "*SequenceVariant*" attributes. This model is a wrapper for the known combinations.

objects = <django_mri.models.managers.sequence_type.SequenceTypeManager object>

sequence_definition_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

sequence_definitions

4.1.4 Serializers

Module contents

Serializers for the app's models.

References

- Django Rest Framework's serializers documentation.

Subpackages

Input

Module contents

Serializers for the *NiftiInput*, *NiftiInputDefinition*, *ScanInput*, and *ScanInputDefinition* models.

Submodules

django_mri.serializers.input.nifti_input module

Definition of the *NiftiInputSerializer* class.

class django_mri.serializers.input.nifti_input.**NiftiInputSerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)

Bases: `rest_framework.serializers.ModelSerializer`

Serializer for the *NiftiInput* model.

value = None
Hyperlink to the actual *django_mri.models.nifti.NIfTI* instance that was used.

django_mri.serializers.input.nifti_input_definition module

Definition of the *NiftiInputDefinitionSerializer* class.

class django_mri.serializers.input.nifti_input_definition.**NiftiInputDefinitionSerializer**(*in*
da
're
****

Bases: rest_framework.serializers.ModelSerializer

Serializer for the *NiftiInputDefinition* model.

django_mri.serializers.input.scan_input module

Definition of the *ScanInputSerializer* class.

class django_mri.serializers.input.scan_input.**ScanInputSerializer**(*instance=None*,
data=<class
'rest_framework.fields.empty'>,
***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

Serializer for the *ScanInput* model.

value = None
Hyperlink to the actual *django_mri.models.scan.Scan* instance that was used.

django_mri.serializers.input.scan_input_definition module

Definition of the *ScanInputDefinitionSerializer* class.

class django_mri.serializers.input.scan_input_definition.**ScanInputDefinitionSerializer**(*insta*
data=
*'rest_*
***kw*

Bases: rest_framework.serializers.ModelSerializer

Serializer for the *ScanInputDefinition* model.

Output

Module contents

Serializer for the *NiftiOutput* and *NiftiOutputDefinition* models.

Submodules

django_mri.serializers.output.nifti_output module

Definition of the *NiftiOutputSerializer* class.

class django_mri.serializers.output.nifti_output.**NiftiOutputSerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

Serializer for the *NiftiOutput* model.

django_mri.serializers.output.nifti_output_definition module

Definition of the *NiftiOutputDefinitionSerializer* class.

class django_mri.serializers.output.nifti_output_definition.**NiftiOutputDefinitionSerializer**

Bases: rest_framework.serializers.ModelSerializer

Serializer for the *NiftiOutputDefinition* model.

Submodules

django_mri.serializers.nifti module

Definition of the *NiftiSerializer* class.

class django_mri.serializers.nifti.**NiftiSerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)

Bases: rest_framework.serializers.HyperlinkedModelSerializer

Serializer for the *NIfTI* model.

django_mri.serializers.scan module

Definition of the *ScanSerializer* class.

class django_mri.serializers.scan.**ScanSerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)

Bases: rest_framework.serializers.HyperlinkedModelSerializer

Serializer class for the User model.

References

- https://www.django-rest-framework.org/api-guide/serializers/

create(*data: dict*)

Gets or creates an instance of the *Scan* model based on the provided data.

Parameters **data** (*dict*) – Instance data

Returns Matching scan

Return type *Scan*

django_mri.serializers.sequence_type module

class django_mri.serializers.sequence_type.**SequenceTypeSerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)

Bases: rest_framework.serializers.HyperlinkedModelSerializer

4.1.5 Utilities

Module contents

General app utilities.

Submodules

django_mri.utils.bids module

Definition of the *Bids* class.

class django_mri.utils.bids.**Bids**(*scan*)

Bases: object

A class to compose BIDS-appropriate paths for usage by dcm2niix In short, standard template for BIDS-appropriate path is: *sub-<label>/<data_type>/sub-<label>_<modality_label>*

References

- The BIDS Specification.

DATASET_DESCRIPTION_FILE_NAME = 'dataset_description.json'

PARTICIPANTS_FILE_NAME = 'participants.tsv'

calculate_age(*born: datetime.date*) → float

Returns age by date of birth.

Parameters **born** (*datetime.date*) – Subject's birth date

Returns Subject's age

Return type float

clean_unwanted_files(*bids_path: pathlib.Path*)

Clean irrelevant .bvec and .bval files; Some versions of dcm2niix produce .bvec and .bval for fieldmap images as well as dwi images. Since BIDS specifications do now allow such files under "fmap" data-type, this method deletes them from the relevant directory.

Parameters **bids_path** (*Path*) – Path to the BIDS-compatible directory

compose_bids_path()

Uses parameters extracted by {self.get_data} to compose a BIDS- compatible file path

Returns Full path to a BIDS-compatible file, according to scan's parameters.

Return type pathlib.Path

fix_functional_json (*bids_path: pathlib.Path*)
Add required "TaskName" field to functional scan, as stated in BIDS stucture.

Parameters **bids_path** (*Path*) – Path to the BIDS-compatible directory

References

- BIDS MRI specification

generate_bidsignore (*parent: pathlib.Path*)
Some acquisitions do not conform to BIDS specification (mainly localizers), so we generate a .bidsignore file, pointing to them.

Parameters **parent** (*Path*) – BIDS-compatible directory, underwhich there are "sub-x" directories

References

- BIDS validator specifications

generate_readme (*parent: pathlib.Path*)
It is recommended by BIDS specifications to have a README file at the base of our project, so we create a blank one for further usage.

Parameters **parent** (*Path*) – BIDS-compatible directory, underwhich there are "sub-x" directories

get_data ()
Extracts relevant parameters for BIDS-compatible naming.

Returns

- **parent** (*Path*) – Parent BIDS directory, under which there will be "sub-x" directories
- **data_type** (*str*) – sub-directory under "sub-x". either "anat","func","fmap" or "dwi"
- **modality_label** (*str*) – modality label as described in BIDS specifications. either "dwi", "epi","T1w","FLAIR","bold" or "localizer"
- **task** (*Union[str, None]*) – task name for functional scans. "rest" or None
- **pe_dir** (*Union[str, None]*) – PhaseEncodingDirection for DWI-related images or fieldmap-related images. Either "AP","PA" or None

get_subject_data ()
Extract relevant Scan-related subject's parameters, as stated by BIDS structure.

Returns subject's relevant parameters, sorted by "participant_id", "handedness","age" and "sex" fields

Return type subject_dict[dict]

set_description_json (*parent: pathlib.Path*)
Generates required "dataset_description.json" file, as stated by BIDS structure.

Parameters **parent** (*Path*) – BIDS-compatible directory, underwhich there are "sub-x" directories

References

- BIDS complementary files

set_participant_tsv_and_json (*parent: pathlib.Path*, *subject_dict: dict*)
Generates recommended "participants.tsv" by either copying the template from TEMPLATES_DiR or editing an existing one under {parent} directory.

Parameters

- **parent** (*Path*) – BIDS-compatible directory, underwhich there are "sub-x" directories
- **subject_dict** (*dict*) – Subject's parameters dictionary, containing "participant_id", "handedness","age","sex" fields

References

- BIDS complementary files

class django_mri.utils.bids.**DATA_TYPES**
Bases: enum.Enum

An enumeration.

AP = 'dwi'

PA = 'fmap'

dwi = 'dwi'

flair = 'anat'

fmri = 'func'

ir_epi = 'anat'

ir_epi2 = 'anat'

localizer = 'anat'

mprage = 'anat'

class django_mri.utils.bids.**MODALITY_LABELS**
Bases: enum.Enum

An enumeration.

AP = 'dwi'

PA = 'epi'

dwi = 'dwi'

flair = 'FLAIR'

fmri = 'bold'

ir_epi = 'T1w'

ir_epi2 = 'T1w'

localizer = 'localizer'

mprage = 'T1w'

django_mri.utils.compression module

Definition of the *compress()* and *uncompress()* utility functions.

django_mri.utils.compression.**compress**(*source: pathlib.Path*, *destination: pathlib.Path = None*, *keep_source: bool = True*) → pathlib.Path

Compresses the provided *source* file.

Parameters

- **source** (*Path*) – File to compress
- **destination** (*Path, optional*) – Compressed output file path, by default None
- **keep_source** (*bool, optional*) – Whether to keep the source file or not, by default True

Returns Output file path

Return type Path

django_mri.utils.compression.**uncompress**(*source: pathlib.Path*, *destination: pathlib.Path = None*, *keep_source: bool = True*) → pathlib.Path

Uncompresses the provided (compressed) *source* file.

Parameters

- **source** (*Path*) – File to uncompress
- **destination** (*Path, optional*) – Uncompressed output file path, by default None
- **keep_source** (*bool, optional*) – Whether to keep the source file or not, by default True

Returns Output file path

Return type Path

django_mri.utils.messages module

A module storing strings used to display messages.

django_mri.utils.scan_type module

Definition of the *django_mri.utils.scan_type.ScanType* Enum.

class django_mri.utils.scan_type.**ScanType**

Bases: enum.Enum

Supported scan file formats.

DICOM = 'DICOM'

NIFTI = 'NIfTI'

django_mri.utils.utils module

General app utilites.

django_mri.utils.utils.**DEFAULT_DICOM_DIR_NAME = 'DICOM'**

The name of the subdirectory under the MRI data root in which DICOM files will be saved.

django_mri.utils.utils.**DEFAULT_MRI_DIR_NAME = 'MRI'**
The name of the subdirectory under MEDIA_ROOT in which MRI data will be saved.

django_mri.utils.utils.**DEFAULT_STUDY_GROUP_MODEL = 'research.Group'**
Default identifier for a study group model scans should be related to.

django_mri.utils.utils.**DEFAULT_SUBJECT_MODEL = 'research.Subject'**
Default identifier for a subject model scans should be related to.

django_mri.utils.utils.**get_dicom_root**() → pathlib.Path
Returns the path of the directory in which DICOM data should be saved.

django_mri.utils.utils.**get_group_model**()
Returns the study group model MRI scans should be related to.

Returns Study group model

Return type django.db.models.Model

django_mri.utils.utils.**get_mri_root**() → pathlib.Path
Returns the path of the directory in which MRI data should be saved.

django_mri.utils.utils.**get_subject_model**()
Returns the subject model MRI scans should be related to.

Returns Subject model

Return type django.db.models.Model

4.1.6 Views

Module contents

Submodules

django_mri.views.defaults module

class django_mri.views.defaults.**DefaultsMixin**
Bases: object

Default settings for view authentication, permissions and filtering.

authentication_classes = (<class 'rest_framework.authentication.BasicAuthentication'>,

filter_backends = (<class 'django_filters.rest_framework.backends.DjangoFilterBackend'

permission_classes = (<class 'rest_framework.permissions.IsAuthenticated'>,)

django_mri.views.nifti module

class django_mri.views.nifti.**NiftiViewSet**(***kwargs*)
Bases: *django_mri.views.defaults.DefaultsMixin*, rest_framework.viewsets.ModelViewSet

pagination_class
alias of *django_mri.views.pagination.StandardResultsSetPagination*

queryset

serializer_class
alias of *django_mri.serializers.nifti.NiftiSerializer*

django_mri.views.pagination module

class django_mri.views.pagination.**StandardResultsSetPagination**
Bases: rest_framework.pagination.PageNumberPagination

Default pagination parameters. This didn't work as part of the DefaultsMixin and therefore has to be defined separately in the 'pagination_class' configuration.

page_size = 25

page_size_query_param = 'page_size'

django_mri.views.scan module

class django_mri.views.scan.**ScanViewSet**(***kwargs*)
Bases: *django_mri.views.defaults.DefaultsMixin*, rest_framework.viewsets.ModelViewSet

API endpoint that allows scans to be viewed or edited.

filter_class
alias of *django_mri.filters.scan_filter.ScanFilter*

from_dicom(*request: rest_framework.request.Request*, *series_id: int = None*) → rest_framework.response.Response
Returns scan information from a Series instance without serializing.

Parameters

- **request** – A request from the client.
- **series_id** (*int, optional*) – Series primary key, by default None

Returns Serialized data or messagerequirements

Return type Response

get_queryset() → django.db.models.query.QuerySet
Filter the returned scans according to the studies the requesting user is a collaborator in, unless the user is staff, in which case return all scans.

Returns Scan instances.

Return type QuerySet

ordering_fields = ('id', 'description', 'number', 'created', 'echo_time', 'inversion_t

pagination_class
alias of *django_mri.views.pagination.StandardResultsSetPagination*

plot(*request: rest_framework.request.Request*, *pk: int = None*) → rest_framework.response.Response

preview_script(*request: rest_framework.request.Request*, *pk: int = None*) → rest_framework.response.Response

queryset

search_fields = ('id', 'description', 'number', 'created', 'scan_time', 'echo_time', '

serializer_class
alias of *django_mri.serializers.scan.ScanSerializer*

django_mri.views.sequence_type module

class django_mri.views.sequence_type.**SequenceTypeViewSet**(***kwargs*)
Bases: *django_mri.views.defaults.DefaultsMixin*, rest_framework.viewsets.ModelViewSet

pagination_class
alias of *django_mri.views.pagination.StandardResultsSetPagination*

queryset

serializer_class
alias of *django_mri.serializers.sequence_type.SequenceTypeSerializer*

django_mri.views.utils module

django_mri.views.utils.**fix_bokeh_script**(*html: str*, *destination_id: str = 'bk-app'*) → str

4.2 Submodules

4.3 django_mri.admin module

Registers various admin models to generate the app's admin site interface.

References

- The Django admin site

class django_mri.admin.**ScanAdmin**(*model*, *admin_site*)
Bases: django.contrib.admin.options.ModelAdmin

Adds the *Scan* to the admin interface.

list_display = ('id', 'subject', 'time', 'number', 'description')
Fields displayed on the change list page of the admin.

media

ordering = ('subject', 'time', 'number')
List ordering in the Django admin views.

4.4 django_mri.apps module

Definition of the *DjangoMriConfig* class.

References

- Django applications

class django_mri.apps.**DjangoMriConfig**(*app_name*, *app_module*)
Bases: django.apps.config.AppConfig

django_mri app configuration.

References

- AppConfig attributes

name = 'django_mri'
Full Python path to the application.

ready()
Loads the app's signals.

References

- ready()

verbose_name = 'MRI Data Management'
Human-readable name for the application.

4.5 django_mri.signals module

Signal receivers.

References

- Signals

django_mri.signals.**scan_post_save_receiver**(*sender: django.db.models.base.Model*, *instance: django_mri.models.scan.Scan*, *created: bool*, ***kwargs*) → None
Creates a new subject automatically if a subject was not assigned and a DICOM series is accessible by extracting the Patient information.

Parameters

- **sender** (*Model*) – The *Scan* model
- **instance** (Scan) – Scan instance
- **created** (*bool*) – Whether the scan instance was created or not

django_mri.signals.**series_post_save_receiver**(*sender: django.db.models.base.Model*, *instance: django_dicom.models.series.Series*, *created: bool*, ***kwargs*) → None
Create a new *Scan* for any created DICOM Series in case one doesn't exist.

Parameters

- **sender** (*Model*) – The Series model
- **instance** (*Series*) – Series instance
- **created** (*bool*) – Whether the series instance was created or not

4.6 django_mri.urls module

The app's URLs configuration.

References

- URL dispatcher

django_mri.urls.**path**(*route*, *view*, *kwargs=None*, *name=None*, *, *Pattern=<class 'django.urls.resolvers.RoutePattern'>*)

django_mri.urls.**router = <rest_framework.routers.DefaultRouter object>**
Automatic URL routing using Django REST Framework.

References

- Routers

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index

D

E

F

G

I

J

L

M

N

O

P

Q

R

S

T

U

V

W