# FMRIF Tools Documentation

***Release 0.1.0a2***

**Jan Varada**

**Aug 09, 2018**

# Contents

# CHAPTER 1

# Introduction

The **fmrif_tools** package contains a set of python utilities to aid in the processing and manipulation of scanner data stored in the Gold and Oxygen archive systems at the Functional Magnetic Resonance Imaging Facility (FMRIF) at the National Institutes of Health. The utilities run under either Python 2.7+ or Python 3.5+.

**Current utilities:**

- **oxy2bids - A utility to convert DICOM scans downloaded from Gold and Oxygen into a BIDS compatible** directory structure
- **bidsmapper - Creates a DICOM to BIDS mapping file, from supplied Gold or Oxygen archives. (Note: This** only creates the mapping file, it does not use the mapping file to carry out the conversion. Use **oxy2bids** for that.)
- **dcmexplorer - Takes in a list of DICOM tags and searches through supplied Gold or Oxygen DICOM files,** and outputs the values of the tags to a csv file.
- **biounpacker - Extracts physiological data from Biopac and saves it in 1D file format, compatible with** AFNI.

For more information, or assistance, contact the lead developer at:

```
jan.varada -at- nih -dot- gov
```

Please submit any bug reports or feature requests at Report bugs or Feature Requests.

CHAPTER 2

# Installation

## 2.1 Setup

1. **Log into radon with your NIH username (must be in the internal network or VPN),** `ssh <username>@radon.nimh.nih.gov`
2. **Add the following to your .bashrc file,** `export PATH="/opt/dcm2niix/bin:/opt/anaconda/anaconda3/bin:$PATH"`
3. Restart the shell (or reload the .bashrc file)
4. **Launch the fmrif_tools virtual environment,** `source activate fmrif_tools`
5. **Create the output directory in which you want to store the results,** `mkdir <output directory>`
6. **The utilities are now ready to use from the command line (or terminal), e.g.,** `oxy2bids [options] ...`
7. **To exit the virtual environment once you are done using the utilities, type** `source deactivate`

## 2.2 Oxygen/Gold Input Data Directory Structure

- **At a minimum, a folder containing the Oxygen/Gold files to be parsed and extracted is needed.**
  - The files might be a mix of compressed Oxygen/Gold .tgz files, or the resulting uncompressed files.
  - If DICOM directory has uncompressed files, the utility expects the file structure and naming from Oxygen/Gold has not been changed.
    - Example:

      Suppose we want to create a BIDS hierarchy from two Oxygen files, DOE_JOHN-12345-20170101-56789-DICOM.tgz and DOE_JANE-23456-20160101-56789-DICOM.tgz. The DOE_JANE... file has been uncompressed. The files should be located in a common directory, say *oxygen_data*, and the filesystem hierarchy should look as follows:

```
oxygen_data/
    DOE_JOHN-12345-20170101-56789-DICOM.tgz
    DOE_JANE-23456/
        20160101-56789/
            mr_0001/
                *.dcm
```

CHAPTER 3

# bidsmapper

## 3.1 Introduction

**bidsmapper** is a python utility that takes in a series of DICOM scans from Oxygen, and generates an intermediate csv file containing a mapping of DICOM scans to BIDS hierarchy.

## 3.2 Basic Command and Flags

- **The basic command structure of bidsmapper is as follows,** `bidsmapper [options] <dicom data directory> <output_directory>`
- The following **options** are allowed:

  **–biopac_dir** Directory where the biopac data indicated in the mapping file is present. At the moment the utility is not capable of matching biopac files to exams/scans automatically.

  **–config** Custom configuration file containing bids tags, dicom tags, or both.

  **–nthreads** Number of threads the program should use when parsing the DICOM files and generating the BIDS dataset.

  **–debug** Outputs useful information for debugging to the log and console.
- For more information on how to combine these flags, see the supported use cases in the following sections.

CHAPTER 4

# oxy2bids

## 4.1 Introduction

**oxy2bids** is a python utility that takes in a series of DICOM scans from Oxygen and =converts the scans to a BIDS hierarchy specified by a DICOM to BIDS mapping file. If a mapping file is not provided, it will attempt to automatically generate one using the **bidsmapper** utility. It uses **dcm2niix** behinds the scenes to convert the DICOM scans to NIFTI format.

## 4.2 Basic Command and Flags

- **The basic command structure of oxy2bids is as follows,** `oxy2bids [options] <dicom data directory> <output_directory>`
- The following **options** are allowed:

  **–bids_dir** Existing BIDS directory. If not specified, a directory called **bids_data_<timestamp>** will be created in the output directory.

  **–bids_map** Path to an existing DICOM to BIDS mapping file. If not specified, the utility will attempt to create one based on either the default mapping tags or any tags specified by the user in the config file.

  **–biopac_dir** Directory where the biopac data indicated in the mapping file is present. At the moment the utility is not capable of matching biopac files to exams/scans automatically.

  **–config** Custom configuration file containing bids tags, dicom tags, or both.

  **–overwrite** If files exist in BIDS data folder, overwrite them. **Note: Not implemented yet.** Default: False.

  **–nthreads** Number of threads the program should use when parsing the DICOM files and generating the BIDS dataset.

  **–debug** Outputs useful information for debugging to the log and console.

- For more information on how to combine these flags, see the supported use cases in the following sections.

CHAPTER 5

# Use Cases

## 5.1 Use Case 1 (Recommended) - Inspect Intermediate Mapping File

1. **Run bidsmapper to generate a mapping file,** `bidsmapper <dicom data directory> <output directory>`
2. Inspect the resulting csv file (will be stored in the output directory, and named bids_map_<datetime>.csv) using your favorite CSV viewer (i.e. R, Pandas, Excel). If any changes are needed, apply the changes and save the file.
3. **Use the mapping file to perform the conversion to BIDS,** `oxy2bids --bids_map <output dir>/bids_map_<datetime>.csv <dicom data dir> <output directory>`
4. Upon completion, data will be stored in <output_dir>/bids_data_<datetime>/.

## 5.2 Use Case 2 - Run oxy2bids Without Inspecting Mapping File

1. To generate a BIDS directory structure without inspecting the intermediate mapping file, simply call **oxy2bids** without specifying a **bids_map**,

   ```
   oxy2bids <dicom data directory> <output directory>
   ```

2. Upon completion, data will be stored in <output_dir>/bids_data_<datetime>/.

CHAPTER 6

# Advanced Usage

## 6.1 Default DICOM Tags

The following DICOM tags can be searched for BIDS tags by default:

- **Study Date** - (0x0008, 0x0020)
- **Station Name** - (0x0008, 0x1010)
- **Manufacturer** - (0x0008, 0x0070)
- **Series Description** - (0x0008, 0x103e)
- **Sequence Name** - (0x0019, 0x109c) or (0x0024)

By default, the utilities will look for bids tags in the **series description** field.

## 6.2 Default BIDS Tags

The following tags should be specified in the **series description** field of the DICOM header.

**Anatomical Scans**

- T1w
- T2w
- T1map
- T2map
- FLAIR
- FLASH
- PD
- PDT2

- inplaneT1
- inplaneT2
- angio
- defacemask
- SWImageandphase

**Functional Scans**

- bold
- bold sbref (can be in any order)

Functional scans may also include a **task** tag that will be parsed to extract the corresponding task name, and it should be specified as follows:

- **task-<task name>**, where **<task name>** is the name of the task corresponding to that series.

**Diffusion Weighted Scans**

- dwi
- dwi sbref (can be in any order)

**Fieldmap Scans**

Due to the multitude of fieldmap scan sequences and possible series arrangements out of the scanners, no tags are supported by default.

Please see the advanced section of this document for a guide on how to specify custom tags for your fieldmap scans.

**Other supported tags**

**Anatomical**, **functional**, and **diffusion weighted** scans also support an acquisitions label tag. From the BIDS Spec (v1.0.1): "...the user may use to distinguish a different set of parameters used for acquiring the same modality". You can specify this tag as follows:

- **acq-<label>** where **<label>** is a string of lower and uppercase letters.

**Anatomical** and **functional** scans support a reconstruction label tag. From the BIDS Spec (v1.0.1): "...can be used to distinguish different reconstruction algorithms". You can specify tag as follows:

- **rec-<label>** where **<label>** is a string of lower and uppercase letters.

## 6.3 Defining custom tags in DICOM header

- Custom BIDS and DICOM tags can be specified in a config file, and passed to the utilities at runtime.
- Passing a custom section in a config file overrides that entire section of the default config file. Thus, if you want to add tags but keep the original ones, it is recommended you obtain the default config file, add the tags to it, and submit that as the custom config file.
- **The default config file can be downloaded from,** `https://raw.githubusercontent.com/nih-fmrif/fmrif_tools/debug_branch/common_utils/data/config.json`
- Note the config file is a JSON-formatted file with the following high level structure,

```
{
    "DICOM_TAGS": { ... },
    "BIDS_TAGS": { ... }
}
```

- The DICOM_TAGS key should consist of a series of dicom tag aliases for a particular DICOM field, followed either by a string containing the hexadecimal values for the DICOM field (separated by a comma), or a list of such string tuples, that is,

```
"DICOM_TAGS": {
    "study_date": "0x0008,0x0020",
    "sequence_name": ["0x0019,0x109c", "0x0018,0x0024"]
}
```

- At a first pass, the BIDS_TAGS key contains bids tags organized by modality,

```
"BIDS_TAGS": {
    "anat": [ ... ],
    "func": [ ... ],
    ...
}
```

- The supported modalities are **anat**, **func**, **dwi**, and **fmap**.
- Each modality contains a list of dictionaries representing a bids tag schema, e.g

```
"anat": [
    {
        "bids_modality": "T1w",
        "include": [],
        "exclude": [],
        "acq": [],
        "rec": []
    }
}
```

- The **bids_modality**, the **include** field is mandatory. The **exclude** field might be an empty list.
- The **include** and **exclude** lists consists of tuples where the first element is a dicom tag alias that has been specified in the DICOM_TAGS key, which represents the DICOM tag in which the program will look for the match parameter, and the second element represents the string to match (or a regular expression), e.g.,

```
{
    "bids_modality": "T1w",
    "include": [
        ["series_description", "T1w"]
    ],
    "exclude": [
        ["sequence_name", "re::^T2w$"]
    ]
}
```

- If a regular expression is specified, it must be prefixed with **re::**.
- If a normal string is passed as the term to match, it will be tested against the value of the DICOM field in the following manner (search is case-insensitive),

```
if search_term in dicom_field.value:
```

- That is, it will return a match if the search term is found as a substring of at any position of the value of the DICOM field.
- If the search term is a regular expression, it will be a match if the DICOM field value satisfies the regular expression.

CHAPTER 7

# Usage with MRIQC

- The resulting top-level bids directory can be used with any tool that recognizes BIDS structured data.
- **mriqc** has been installed as a docker container on **radon**.
- It is installed as the docker container **poldracklab/mriqc**.
- **To use, see the instructions at,** `http://mriqc.readthedocs.io/en/stable/docker.html`

CHAPTER 8

# License