bidspm

Release v3.1.0dev

the bidspm pipeline dev team

May 08, 2024

CONTENT

CHAPTER

ONE

BIDSPM

This is a Matlab / Octave toolbox to perform MRI data analysis on a BIDS data set using SPM12.

1.1 Installation

1.1.1 From docker hub

```
docker pull cpplab/bidspm:latest
```

1.1.2 From the source

In a terminal or a git bash prompt, type:

```
git clone --recurse-submodules https://github.com/cpp-lln-lab/bidspm.git
```

To start using bidspm, you just need to initialize it for this MATLAB / Octave session with:

```
bidspm()
```

Please see our documentation for more info.

1.1.3 Installing the Command line Interface (CLI)

If you want to use the BIDS app python based CLI of bidspm.

You need to

- python3
- pip

If you are using MATLAB, you need to edit the file `src/matlab.py`, so that it returns the fullpath to the MATLAB executable on your computer.

You can then install the bidspm CLI from within the `bidspm` folder with:

```
pip install .
```

You can then type the following to see which command you have access to:

```
bidspm --help
```

1.1.4 BIDS validation

After installing bidspm python package, you can get access to extra validation options.

1.1.5 BIDS stats model validation

Please see the documentation

1.1.6 BIDS dataset validation

To run the bids-validator when running bidspm, you will need:

- node.js and npm
- the bidspm python CLI (see above)

You can then install:

- the bids validator

by running from the command line in the root folder of the repository:

```
make install
```

or

```
npm install -g bids-validator
```

1.2 Usage

For some of its functionality bidspm has a BIDS app like API.

See this page for more information.

But in brief they are of the form:

```
bidspm(bids_dir, output_dir, ...
       'analysis_level', ...
       'action', 'what_to_do')
```

1.2.1 Creating a default BIDS statistical model

Use a MATLAB / Octave script with:

```
% path to your raw BIDS dataset
bids_dir = path_of_raw_bids_dataset;

% where you want to save the model
output_dir = path_where_the_output_should_go;

tasks_to_include_in_model = {'task1', 'task2', 'task3'};

% for example 'MNI152NLin2009cAsym'
space_to_include_in_model = {'spaceName'};

bidspm(bids_dir, output_dir, 'dataset', ...
        'action', 'default_model', ...
        'task', tasks_to_include_in_model, ...
        'space', space_to_include_in_model)
```

1.2.2 GLM

Use a MATLAB / Octave script with:

```
% path to your raw BIDS dataset
bids_dir = path_of_raw_bids_dataset;

% where you want to save the model
output_dir = path_where_the_output_should_go;

preproc_dir = path_to_preprocessed_dataset; % for example fmriprep output

model_file = path_to_bids_stats_model_json_file;

subject_label = '01';

bidspm(bids_dir, output_dir, 'subject', ...
        'participant_label', {subject_label}, ...
        'action', 'stats', ...
        'preproc_dir', preproc_dir, ...
        'model_file', model_file)
```

1.2.3 Preprocessing

```
bids_dir = path_to_raw_bids_dataset;
output_dir = path_to_where_the_output_should_go;

subject_label = '01';

bidspm(bids_dir, output_dir, 'subject', ...
        'participant_label', {subject_label}, ...
```

(continues on next page)

(continued from previous page)

```
        'action', 'preprocess', ...
        'task', {'yourTask'})
```

1.3 Features

1.3.1 Statistics

The model specification are set up using the BIDS stats model and can be used to perform:

- whole GLM at the subject level
- whole brain GLM at the group level à la SPM (meaning using a summary statistics approach).
- ROI based GLM (using marsbar)
- model selection (with the MACS toolbox)

1.3.2 Preprocessing

If your data is fairly "typical" (for example whole brain coverage functional data with one associated anatomical scan for each subject), you might be better off running fmriprep on your data.

If you have more exotic data that cannot be handled well by fmriprep then bidspm has some automated workflows to perform amongst other things:

- remove dummies
- slice timing correction
- spatial preprocessing:
  - realignment OR realignm and unwarp
  - coregistration `func` to `anat`,
  - `anat` segmentation and skull stripping
  - (optional) normalization to SPM's MNI space
- smoothing
- fieldmaps processing and voxel displacement map creation (work in progress)

All (well almost all) preprocessed outputs are saved as BIDS derivatives with BIDS compliant filenames.

1.3.3 Quality control:

- anatomical data (work in progress)
- functional data (work in progress)
- GLM auto-correlation check

Please see our documentation for more info.

1.4 Citation

```
@software{bidspm,
  author = {Gau, Rémi and Barilari, Marco and Battal, Ceren and Rezk, Mohamed and␣
↪Collignon, Olivier and Gurtubay, Ane and Falagiarda, Federica and MacLean, Michèle and␣
↪Cerpelloni, Filippo and Shahzad, Iqra and Nunes, Márcia and Caron-Guyon, Jeanne and␣
↪Chouinard-Leclaire, Christine and Yang, Ying and Mattioni, Stefania and Van Audenhaege,
↪ Alice and Matuszewski, Jacek},
  license = {GPL-3.0},
  title = {{bidspm}},
  url = {https://github.com/cpp-lln-lab/bidspm},
  version = {3.1.0}
}
```

1.4.1 Posters

- OHBM 2023

1.5 Contributors

Thanks goes to these wonderful people (emoji key):

This project follows the all-contributors specification. Contributions of any kind welcome!

CHAPTER

TWO

INSTALLATION

2.1 Dependencies

This SPM toolbox runs with Matlab and Octave.

Dependencies	Minimum required	Used for testing in CI
MATLAB	2014	2022a on Ubuntu 22.04, Windowns and MacOS
Octave	6.4.0	6.4.0 on Ubuntu 22.04
SPM12	7219	7771

Some functionalities require some extra SPM toolbox to work: for example the ALI toolbox for brain lesion segmentation.

2.1.1 Octave compatibility

The following features do not yet work with Octave:

- *anatQA()*
- slice_display toolbox

Not (yet) tested with Octave:

- MACS toolbox workflow for model selection
- ALI toolbox workflow for model selection

2.2 Installation

If you are only going to use this toolbox for a new analysis and you are not planning to edit the code base of bidspm itself, we STRONGLY suggest you use this template repository to create a new project with a basic structure of folders and with the bidspm code already set up.

Otherwise you can clone the repo with all its dependencies with the following git command:

```
git clone --recurse-submodules https://github.com/cpp-lln-lab/bidspm.git
```

If you need the latest development, then you must clone from the `dev` branch:

2.3 Initialization

Warning: In general DO NOT ADD bidspm PERMANENTLY to your MATLAB / Octave path.

You just need to initialize for a given session with:

```
bidspm()
```

This will add all the required folders to the path.

You can also remove bidspm from the path with:

```
bidspm uninit
```

2.4 Installation on a computing cluster

For stand alone download https://www.fil.ion.ucl.ac.uk/spm/download/restricted/utopia/

To use SPM docker https://github.com/spm/spm-docker

See FAQ: https://en.wikibooks.org/wiki/SPM/Standalone#Frequently_Asked_Questions

This relies on the fact that SPM and CPM SPM are Octave compatible, so you will be able to run most of bidspm on a high performance cluster (HPC) without having to worry about MATLAB licenses.

Of course this assumes that Octave is available on your HPC.

Note that it should also be possible to precompile with MATLAB all the things you want to run, but this is not shown here.

The pre-requisite steps are described in the example below that shows how to set up bidspm on one of the HPC of the université catholique de Louvain.

1. SSH into the HPC

Assumes that you have set things up properly. For the UCLouvain see the documentation on this website (which has some good info about using HPC in general).

If you have everything set up it should be almost as easy as opening a terminal and typing:

```
ssh lemaitre3
```

2. Get SPM

You can simply clone the latest version of SPM from github with:

```
git clone https://github.com/spm/spm12.git --depth 1
```

3. Load the Octave modules

This first step might be different on your HPC, so you might have to figure out what the equivalent modules are called on your HPC (in the UCLouvain case you can find the relevant module by typing `module spider octave`)

Once you have found the modules load them:

```
module load releases/2018b
module load Octave/4.4.1-foss-2018b
```

4. Recompile SPM for Octave

You need to recompile SPM to make sure it works with Octave. This relies on running the following Make commands:

```
make -C spm12/src PLATFORM=octave distclean
make -C spm12/src PLATFORM=octave
make -C spm12/src PLATFORM=octave install
```

5. Add SPM to the path

In the example below `$` shows when you are in the bash terminal and `octave:1>` shows when you are in the Octave terminal.

Launch Octave:

```
$ octave

GNU Octave, version 4.4.1
Copyright (C) 2018 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type 'warranty'.

Octave was configured for "x86_64-pc-linux-gnu".

Additional information about Octave is available at https://www.octave.org.

Please contribute if you find this software useful.
For more information, visit https://www.octave.org/get-involved.html

Read https://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.
```

Add the SPM12 folder to the path and save the path:

```
octave:1> addpath(fullfile(pwd, 'spm12'))
octave:2> savepath
octave:3> exit
```

5. Install bidspm

As before install and run an initialization:

```
git clone \
  -b dev \
  --recurse-submodules \
  https://github.com/cpp-lln-lab/bidspm.git
```

Warning: There are some warnings thrown during initialization:

```
octave:1> initCppSpm
warning: addpath: /home/users/r/g/rgau/bidspm/lib/spmup/utlilities/home/users/r/g/
↪rgau/bidspm/lib/spm_2_bids: No such file or directory
warning: called from initCppSpm at line 67 column 5
warning: function /home/users/r/g/rgau/bidspm/lib/spmup/external/cubehelix.m shadows␣
↪a core library function
```

```
warning: called from initCppSpm at line 67 column 5
warning: addpath: /home/users/r/g/rgau/bidspm/src/workflows/stats/home/users/r/g/rgau/
↪bidspm/lib/spmup: No such file or directory
```

As well as many warnings of the type:

```
sh: makeinfo: command not found
warning: doc_cache_create: unusable help text found in file 'analyze75info'
```

CHAPTER

THREE

USAGE NOTES

3.1 API

3.2 Command line API

bidspm is a SPM base BIDS app

```
Usage: bidspm [-h] [-v] --action
              {preprocess,smooth,default_model,create_roi,stats,contrasts,results}
              [--verbosity {0,1,2}] [--task TASK [TASK ...]]
              [--space SPACE [SPACE ...]]
              [--ignore {fieldmaps,slicetiming,unwarp,qa,contrasts,transformations,
↪dataset} [{fieldmaps,slicetiming,unwarp,qa,contrasts,transformations,dataset} ...]]
              [--participant_label PARTICIPANT_LABEL [PARTICIPANT_LABEL ...]]
              [--boilerplate_only] [--dry_run]
              [--bids_filter_file BIDS_FILTER_FILE] [--fwhm FWHM]
              [--options OPTIONS] [--skip_validation] [--roi_dir ROI_DIR]
              [--roi_name ROI_NAME [ROI_NAME ...]] [--anat_only]
              [--dummy_scans DUMMY_SCANS]
              [--roi_atlas {neuromorphometrics,wang,anatomy_toobox,visfatlas,hcpex}]
              [--model_file MODEL_FILE] [--preproc_dir PREPROC_DIR]
              [--keep_residuals] [--concatenate] [--design_only] [--roi_based]
              bids_dir output_dir {subject,dataset}
```

3.2.1 Positional Arguments

bids_dir Fullpath to the directory with the input dataset formatted according to the BIDS standard.

output_dir Fullpath to the directory where the output files will be stored.

analysis_level Possible choices: subject, dataset

Level of the analysis that will be performed. Multiple participant level analyses can be run independently (in parallel) using the same `output_dir`.

3.2.2 Named Arguments

-v, --version show program's version number and exit

--action Possible choices: preprocess, smooth, default_model, create_roi, stats, contrasts, results

Action to perform.

--verbosity Possible choices: 0, 1, 2

Verbosity level.

Default: 2

--task Tasks of the input data.

--space Space of the input data.

Default: ['IXI549Space']

--ignore Possible choices: fieldmaps, slicetiming, unwarp, qa, contrasts, transformations, dataset

To specify steps to skip.

--participant_label The label(s) of the participant(s) that should be analyzed. The label corresponds to sub-<participant_label> from the BIDS spec (so it does not include "sub-"). If this parameter is not provided all subjects should be analyzed. Multiple participants can be specified with a space separated list. Can be a regular expression. Example: `'01', '03', '08'`.

--boilerplate_only When set to `true` this will only generate figures describing the raw data, the methods section boilerplate.

Default: False

--dry_run When set to `true` this will generate and save the SPM batches, but not actually run them.

Default: False

--bids_filter_file Fullpath to a JSON file describing custom BIDS input filters.

--fwhm The full width at half maximum of the gaussian kernel to apply to the preprocessed data or to use as inputs for the statistical analysis.

Default: 6.0

--options Path to JSON file containing bidspm options.

--skip_validation To skip BIDS dataset and BIDS stats model validation.

Default: False

--roi_dir Fullpath to the directory with the regions of interest.

--roi_name Name of the roi to create. If the ROI does not exist in the atlas, the list of available ROI will be returned in the error message.

3.2.3 preprocess only arguments

--anat_only If preprocessing should be done only on anatomical data.

Default: False

--dummy_scans Number of dummy scans to remove.

Default: 0

3.2.4 create roi only arguments

--roi_atlas Possible choices: neuromorphometrics, wang, anatomy_toobox, visfatlas, hcpex

Atlas to create the regions of interest from.

Default: "neuromorphometrics"

3.2.5 stats only arguments

--model_file Fullpath to BIDS stats model.

--preproc_dir Fullpath to the directory with the preprocessed data.

--keep_residuals Keep GLM residuals.

Default: False

--concatenate To create 4D image of all the beta and contrast images of the conditions of interest included in the run level design matrix.

Default: False

--design_only To only specify the GLM without estimating it.

Default: False

--roi_based To run stats only in regions of interests.

Default: False

- all parameters use `snake_case`,
- most "invalid" calls simply initialize bidspm.

For a more readable version of this help section, see the online https://bidspm.readthedocs.io/en/latest/usage_notes.html.

CHAPTER

FOUR

CONFIGURATION

4.1 Options

Most of the options you have chosen for your analysis will be set in a variable `opt` an Octave/Matlab structure.

Documentation on the options can be found here: *checkOptions()*.

The content of that structure can be defined:

- at run time in a script or a function
- in a separate json file (that will be loaded with *loadAndCheckOptions()*).

You can find examples of both in the `demos` folder.

See *checkOptions()* for a list of all the options set by that function.

See *bidsResults()* for more details on the options to set to get specific results.

4.2 Defaults

The defaults are handled mostly by those functions:

- *checkOptions()*
- *setDirectories()*
- *defaultResultsStructure()*
- *defaultContrastsStructure()*

4.2.1 spm_my_defaults

Some more SPM options can be set in the *spm_my_defaults()*.

4.2.2 statistics defaults

Note that some of the defaults value may be over-ridden by the content of the `opt` structure but also by the content of your BIDS stats model.

Serial correlation modelisation

Use of FAST [OARW19] and not AR1 for auto-correlation modelisation.

Using FAST does not seem to affect results on time series with "normal" TRs but improves results when using sequences: it is therefore used by default in this pipeline.

See the Software section in the *BIDS stats model* page if you want to use the BIDS stats model to change the serial correlation modelisation.

4.2.3 SPM to BIDS filename conversion

set_spm_2_bids_defaults()

4.2.4 Setting directories

Below are some example on how to specify input and output directories.

Note: It will be easier and make your code more portable, if you use relative path for your directory setting.

For preprocessing

For a given folder structure:

```
my_fmri_project
  ├── code
  │   └── getOptionPreproc.m
  ├── outputs/derivatives
  └── inputs
      └── raw
```

Example content of `getOptionPreproc` file:

```
opt.pipeline.type = 'preproc';

this_dir = fileparts(mfilename('fullpath'));

opt.dir.raw = fullfile(this_dir, '..' 'inputs', 'raw');
opt.dir.derivatives = fullfile(this_dir, '..', 'outputs', 'derivatives');
```

For statistics

To run a GLM, bidspm gets the images and confound time series from a preprocessed derivatives BIDS dataset (from fMRIprep or bidspm) and the `events.tsv` files from a raw BIDS dataset.

For a given folder structure:

```
my_fmri_project
   ├── code
   │   └── getOptionStats.m
   ├── outputs/derivatives
   └── inputs
       ├── fmriprep
       └── raw
```

Example content of `getOptionStats` file:

```
opt.pipeline.type = 'stats';

this_dir = fileparts(mfilename('fullpath'));

opt.dir.raw = fullfile(this_dir, '..', 'inputs', 'raw');
opt.dir.preproc = fullfile(this_dir, '..', 'inputs', 'fmriprep');
opt.dir.derivatives = fullfile(this_dir, '..', 'outputs', 'derivatives');
```

The actual `opt.dir.input` and `opt.dir.output` folders will usually be set automatically when running:

```
opt = checkOptions(opt)
```

But you can set those by hand if you prefer.

4.3 List of defaults

CHAPTER

FIVE

BIDS STATS MODEL

This file allows you to specify the GLM to run and which contrasts to compute.

It follows BIDS statistical model.

This type of JSON file is a bit more complicated than the usual JSON files, you might be acquainted with in BIDS. So make sure you have a read through the JSON 101 page.

Then have a look at the walkthrough that explains how to build a simple model.

5.1 Validate your model

5.1.1 In Visual Studio Code

You can add those lines to the `.vscode/settings.json` of your project to help you validate BIDS stats models as you write them.

```
{
  "json.schemas": [
    {
      "fileMatch": [
        "model-*_smdl.json"
      ],
      "url": "https://bids-standard.github.io/stats-models/BIDSStatsModel.json"
    }
  ],
  "esbonio.sphinx.confDir": ""
}
```

5.1.2 In the browser

Otherwise you can use the online validator and copy paste your model in it.

5.1.3 Using the BIDS stats model python package

Requires python and pip.

From within the bidspm folder open a terminal and install the bidspm package.

```
pip install .
```

Then you can validate your model with the following command.

```
validate_model path_to_your_model_json
```

5.2 Loading and interacting with a BIDS stats model

You can use the `BidsModel` class to create a bids model instance and interact with. This class inherits from bids-matlab `bids.Model` class.

class BidsModel

Property Summary

SPM

content of SPM.mat

Method Summary

getHRFderivatives(*varargin*)

returns the HRF derivatives of a node of a BIDS statistical model

getInclusiveMaskThreshold(*varargin*)

returns the threshold for inclusive masking of subject level GLM node of a BIDS statistical model

getModelMask(*varargin*)

returns the mask of a node of a BIDS statistical model

getSerialCorrelationCorrection(*varargin*)

returns the Serial Correlation Correction of a node of a BIDS statistical model

validateConstrasts()

validate all contrasts spec in the model

There are also extra functions to interact with those models.

- `src.bids_model.getContrastsList()`
- `src.bids_model.getDummyContrastsList()`

5.3 bidspm implementation of the BIDS stats model

bidspm only implements a subset of what is currently theoretically possible with the BIDS stats model.

For example, at the subject level the bidspm can only access variables, that are in the `events.tsv` in that raw dataset or in the `regressors.tsv` or `timeseries.tsv` generated by the preprocessing pipeline.

At the group level, it is only possible to access some variables from the `participants.tsv` file.

5.3.1 Transformation

The `Transformations` object allows you to define what you want to do to some variables, before you put them in the design matrix.

Currently bidspm can only transform variables contained in `events.tsv` files.

It uses bids-matlab transformers to run those transformations. Please see this bids-matlab documentation to know how to use them and call them in your JSON.

You can find a list of the available variables transformations in the bids matlab doc and on the variable-transforms repository

The advantage of these bids-matlab transformers is that they allow you to directly test them on tsv files to quickly see what outcome a series of transformers will produce.

Below is an example on how to subtract 3 seconds from the event onsets of the conditions `motion` listed in the `trial_type` columns of the `events.tsv` file, and put the output in a variable called `motion`.

```
"Transformations": {
    "Transformer": "bidspm",
    "Instructions": [
        {
            "Name": "Subtract",
            "Input": [
              "onset"
            ],
            "Query": "trial_type==motion",
            "Value": 3,
            "Output": [
                "motion"
            ]
        }
    ]
}
```

At the subject level, bidspm can only access apply transformation on the content `events.tsv`.

You can find demo of how to design the transformers for your analysis in the `demos/transformers` folder and also in `demos/ds003717/code/04_transformation.m`.

5.3.2 HRF

For a given `Node`, `Model.X` defines the variables that have to be put in the design matrix.

Here `trans_?` means any of the translation parameters (in this case `trans_x`, `trans_y`, `trans_z`) from the realignment that are stored in `_confounds.tsv` files.

Similarly `*outlier*` means that ANY "scrubbing" regressors containing the word `outlier` created by fMRIprep or bidspm to detect motion outlier or potential dummy scans will be included.

```
"Model": {
    "Type": "glm",
    "X": [
        "motion",
        "static",
        "trans_?",
        "rot_?",
        "*outlier*"
    ],
    "HRF": {
        "Variables": [
            "motion",
            "static"
        ],
        "Model": "spm"
    }
}
```

`HRF` specifies:

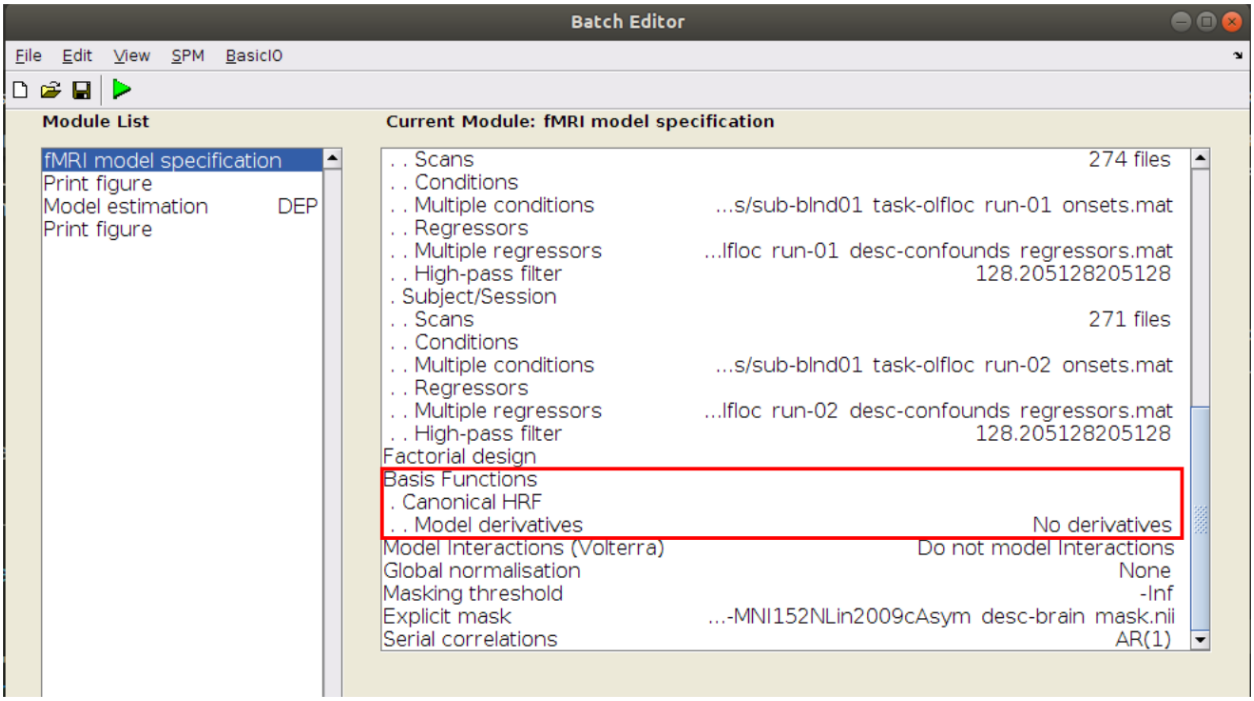
- which variables of `X` have to be convolved
- what HRF model to use to do so.

You can choose from:

- `"spm"`
- `"spm + derivative"`
- `"spm + derivative + dispersion"`

Not yet implemented:

- `"fir"`



5.3.3 Software

By default, bidspm will use SPM's FAST model for the *`SerialCorrelation` model*.

It will also use a value of 0.8 for the `InclusiveMaskingThreshold` to define the implicit inclusive mask that is used by SPM to determine in which voxels the GLM will be estimated (the value is taken from`defaults.mask.thresh` from SPM's defaults).

This corresponds to explicitly setting the following fields in the `Model.Software.SPM` object of a node in the BIDS stats model.

```
{
  "Nodes": [
    {
      "Level": "Run",
      "Name": "run_level",
      "Model": {
        "X": ["trial_type.listening"],
        "HRF": {
          "Variables": ["trial_type.listening"],
          "Model": "spm"
        },
        "Type": "glm",
        "Software": {
          "SPM": {
            "SerialCorrelation": "FAST",
            "InclusiveMaskingThreshold": 0.8
          }
        }
      }
    }
```

(continues on next page)

(continued from previous page)

```
  ]
}
```

These values will explicitly be added to your your default BIDS stats model if you use bidspm 'default_model' action.

```
bidspm(bids_dir, output_dir, 'dataset', ...
        'action', 'default_model')
```

Note that if you wanted to use the `AR(1)` model for the serial correlation and to include all voxels in the implicit mask, you would have to set the following:

```
{
  "Nodes": [
    {
      "Level": "Run",
      "Name": "run_level",
      "Model": {
        "X": ["trial_type.listening"],
        "HRF": {
          "Variables": ["trial_type.listening"],
          "Model": "spm"
        },
        "Type": "glm",
        "Software": {
          "SPM": {
            "SerialCorrelation": "AR(1)",
            "InclusiveMaskingThreshold": "-Inf"
          }
        }
      }
    }
  ]
}
```

Results

It is possible to specify the results you want to view directly the `Model.Software` object of any `Nodes` in the BIDS stats model.

See the help section of the `bidsResults` function for more detail, but here is an example how you could specify it in a JSON.

```
"Model": {
    "Software": {
        "bidspm": {
          "Results": [
              {
                "name": [
                    "contrast_name", "other_contrast_name"
                ],
                "p": 0.05,
                "MC": "FWE",
```

(continues on next page)

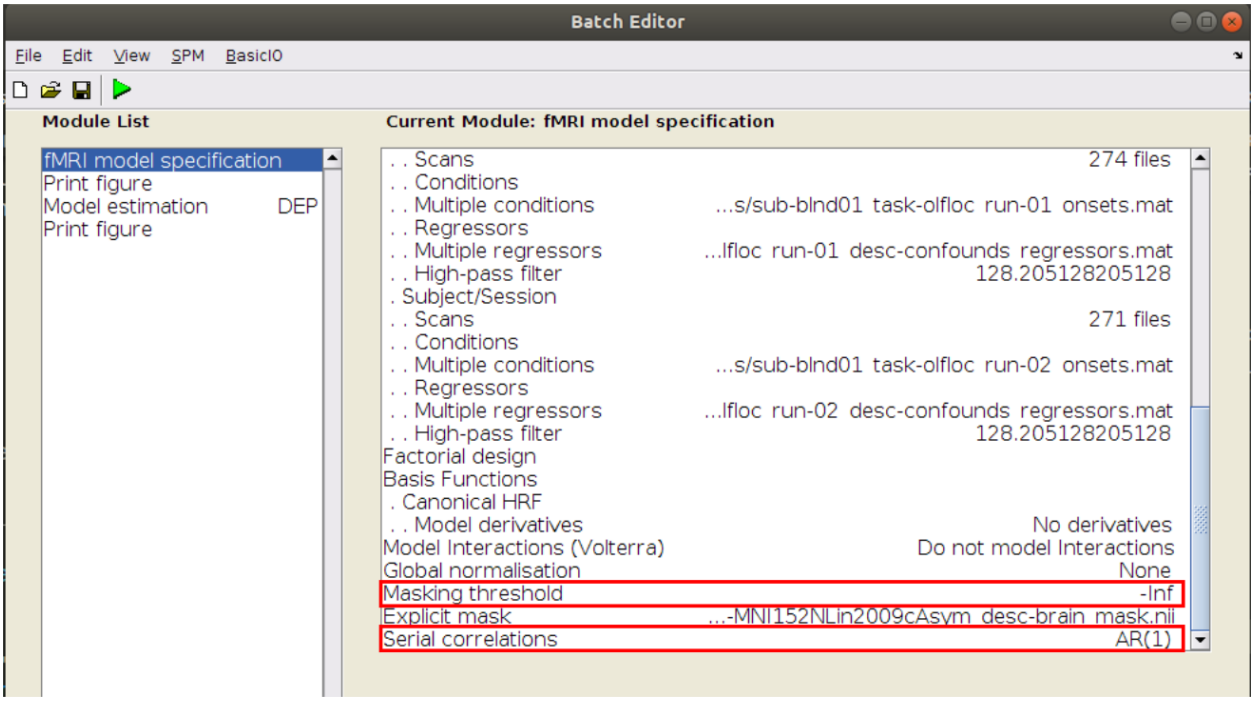


Fig. 1: Corresponding options in SPM batch

(continued from previous page)

```
                "png": true,
                "binary": true,
                "nidm": true,
                "montage": {
                    "do": true,
                    "slices": [
                    -4,
                    0,
                    4,
                    8,
                    16
                    ],
                    "background": {
                    "suffix": "T1w",
                    "desc": "preproc",
                    "modality": "anat"
                    }
                }
              },
              {
                "Description": "Note that you can specify multiple results objects, each␣
↪with different parameters.",
                "name": [
                    "yes_another_contrast_name"
                ],
                "p": 0.01,
                "k": 10,
```

(continues on next page)

(continued from previous page)

```
                "MC": "none",
                "csv": true,
                "atlas": "AAL"
              }
          ]
        }
    }
}
```

5.3.4 Contrasts

Run level

To stay close to the way most SPM users are familiar with, all runs are analyzed in one single GLM.

Contrasts are the run level that are either specified using `DummyContrasts` or `Contrasts` will be computed and will have the run number appended to their name in the SPM gui as shown in *Contrast for run 1* and *Contrast for run 2*.

```
{
  "Level": "Run",
  "Name": "run_level",
  "X": [
    "olfid_eucalyptus_left",
    "olfid_eucalyptus_right",
    "olfid_almond_left",
    "olfid_almond_right",
    "olfloc_eucalyptus_left",
    "olfloc_eucalyptus_right",
    "olfloc_almond_left",
    "olfloc_almond_right",
    "resp_03",
    "resp_12",
    1
  ],
  "DummyContrasts": {
    "Contrasts": [
      "olfid_eucalyptus_left",
      "olfid_eucalyptus_right",
      "olfid_almond_left",
      "olfid_almond_right",
      "olfloc_eucalyptus_left",
      "olfloc_eucalyptus_right",
      "olfloc_almond_left",
      "olfloc_almond_right"
    ],
    "Test": "t"
  },
  "Contrasts": [
    {
      "Name": "olfid",
      "ConditionList": [
```

(continues on next page)

(continued from previous page)

```
        "olfid_eucalyptus_left",
        "olfid_eucalyptus_right",
        "olfid_almond_left",
        "olfid_almond_right"
      ],
      "Weights": [
        1,
        1,
        1,
        1
      ],
      "Test": "t"
    }
  ]
}
```

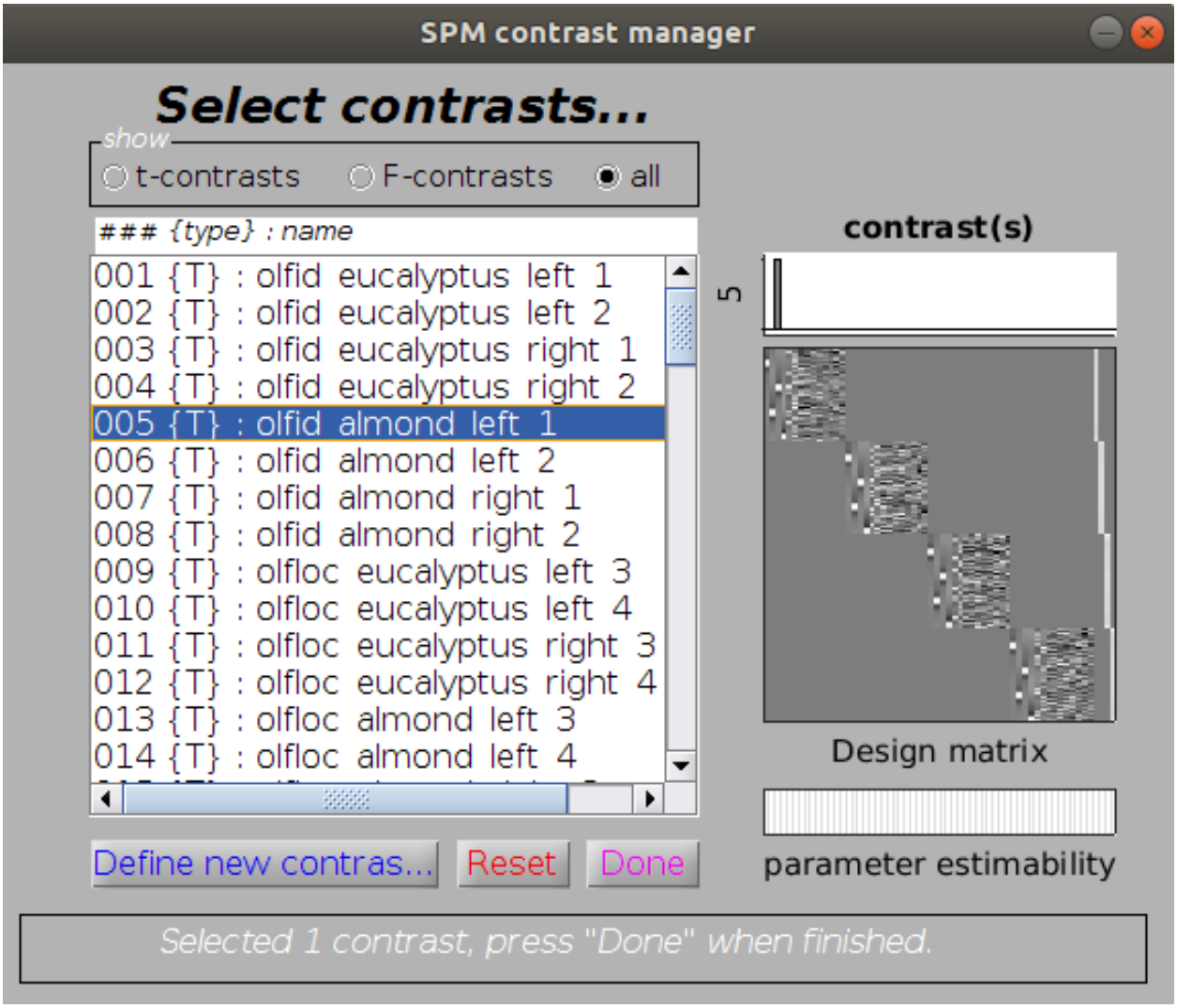


Fig. 2: Contrast for run 1

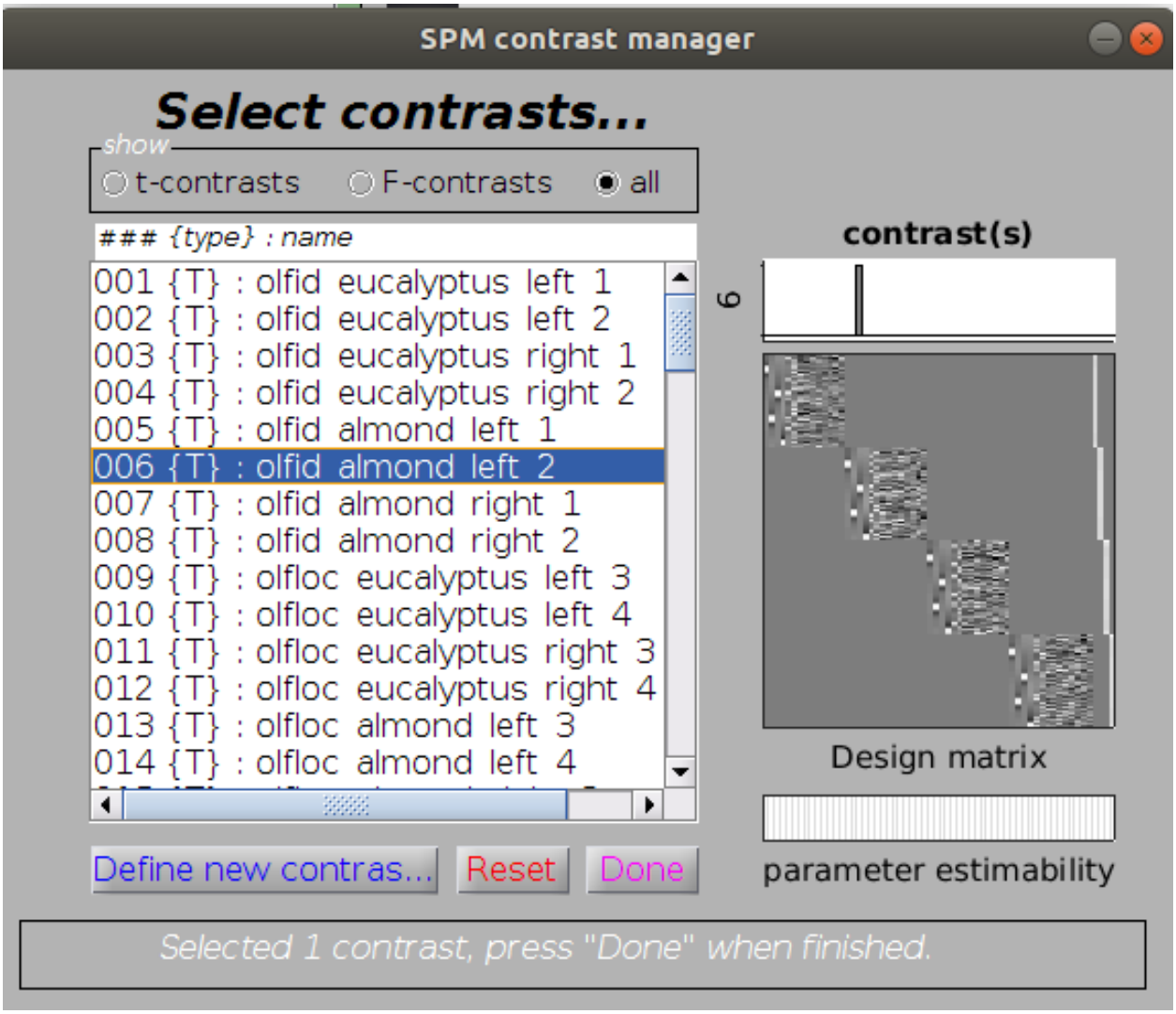


Fig. 3: Contrast for run 2

Subject level

At the moment the only type of model supported at the run level is averaging of run level contrasts.

```
{
  "Level": "Subject",
  "Name": "subject_level",
  "Description": "Only averaging at the subject level is supported for now.",
  "GroupBy": [
    "contrast",
    "subject"
  ],
  "Model": {
    "X": [
      1
    ],
    "Type": "glm"
  },
  "DummyContrasts": {
    "Test": "t"
  }
}
```

5.4 Dataset level

At the moment only, the only type of models that are supported are:

- one sample t-test: averaging across all subjects

```
{
  "Level": "Dataset",
  "Name": "dataset_level",
  "GroupBy": [
    "contrast"
  ],
  "Model": {
    "X": [
      1
    ],
    "Type": "glm"
  },
  "DummyContrasts": {
    "Test": "t"
  }
}
```

- one sample t-test: averaging across all subjects of a specific group

```
{
  "Level": "Dataset",
  "Name": "within_group",
  "Description": "one sample t-test for each group",
```

(continues on next page)

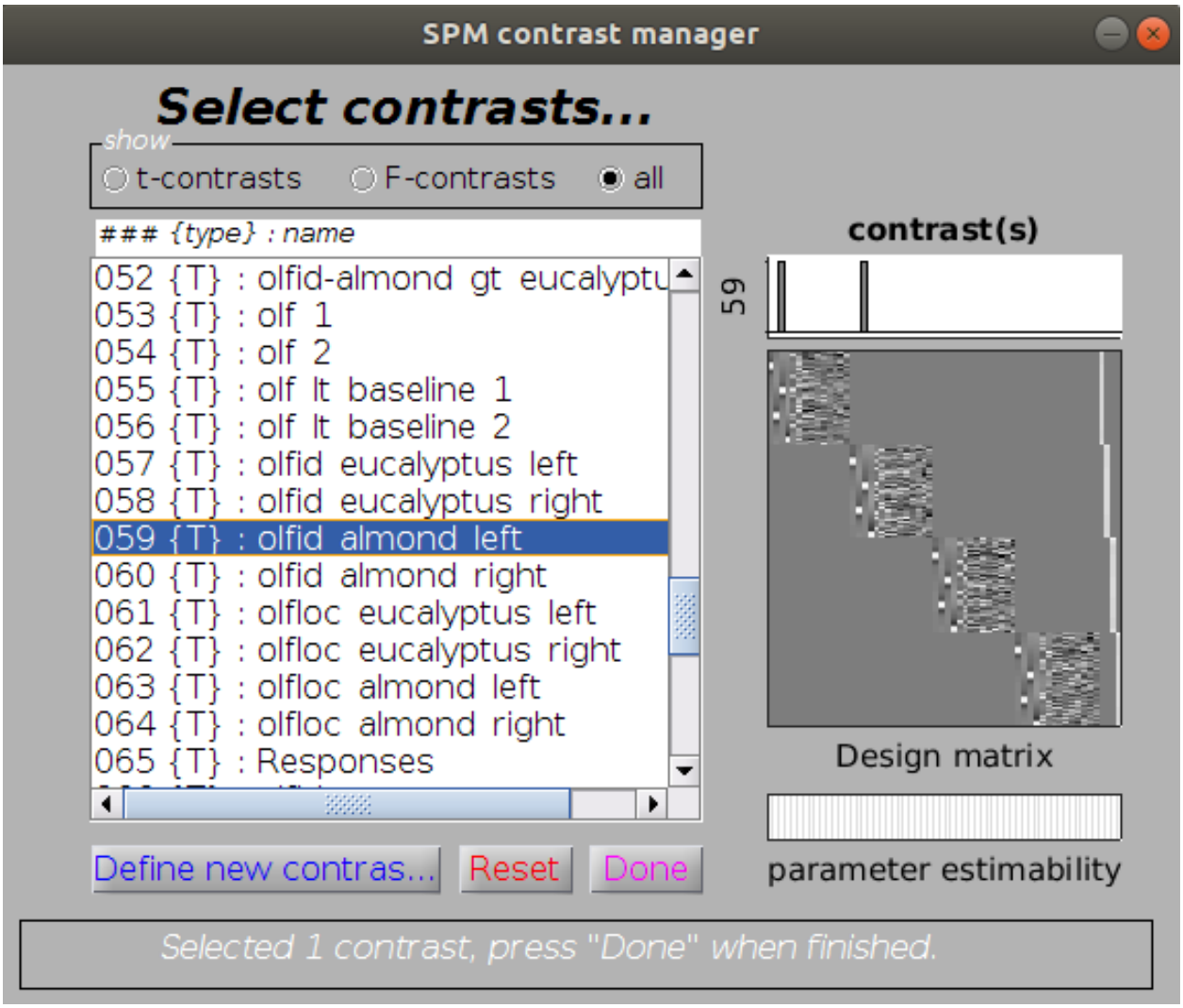


Fig. 4: Subject level contrast averaging beta of run 1 and 2

(continued from previous page)

```
  "GroupBy": [
    "contrast",
    "Group"
  ],
  "Model": {
    "Type": "glm",
    "X": [
      1
    ]
  },
  "DummyContrasts": {
    "Test": "t"
  }
}
```

- 2 samples t-test: comparing 2 groups

At the moment this can only be based on how participants are allocated to a group based on a `group` or `Group` column in the `participants.tsv` of in the raw dataset.

```
{
  "Level": "Dataset",
  "Name": "between_groups",
  "Description": "2 sample t-test between groups",
  "GroupBy": [
    "contrast"
  ],
  "Model": {
    "Type": "glm",
    "X": [
      1,
      "group"
    ]
  },
  "Contrasts": [
    {
      "Name": "blind_gt_control",
      "ConditionList": [
        "Group.blind",
        "Group.control"
      ],
      "Weights": [
        1,
        -1
      ],
      "Test": "t"
    }
  ]
}
```

5.4.1 Method section

It is possible to write a draft of method section based on a BIDS statistical model.

```
opt.model.file = fullfile(pwd, ...
                          'models', ...
                          'model-faceRepetition_smdl.json');
opt.fwhm.contrast = 0;
opt = checkOptions(opt);

opt.designType = 'block';

outputFile = boilerplate(opt, ...
                         'outputPath', pwd, ...
                         'pipelineType', 'stats');
```

```
## fMRI statistical analysis

The fMRI data were analysed with bidspm (v2.2.0; https://github.com/cpp-lln-lab/bidspm;␣
↪DOI: https://doi.org/10.5281/zenodo.3554331)
using statistical parametric mapping
(SPM12 - 7771; Wellcome Center for Neuroimaging, London, UK;
https://www.fil.ion.ucl.ac.uk/spm; RRID:SCR_007037)
using MATLAB 9.4.0.813654 (R2018a)
on a unix computer (Linux version 5.15.0-53-generic (build@lcy02-amd64-047) (gcc (Ubuntu␣
↪11.2.0-19ubuntu1) 11.2.0, GNU ld (GNU Binutils for Ubuntu) 2.38) #59-Ubuntu SMP Mon␣
↪Oct 17 18:53:30 UTC 2022
).

The input data were the preprocessed BOLD images in IXI549Space space for the task "␣
↪facerepetition ".

### Run / subject level analysis

At the subject level, we performed a mass univariate analysis with a linear
regression at each voxel of the brain, using generalized least squares with a
global  AR(1)  model to account for temporal auto-correlation
 and a drift fit with discrete cosine transform basis ( 128 seconds cut-off).

Image intensity scaling was done run-wide before statistical modeling such that
the mean image would have a mean intracerebral intensity of 100.

We modeled the fMRI experiment in a  event  design with regressors
entered into the run-specific design matrix. The onsets
were convolved with SPM canonical hemodynamic response function (HRF)
 and its temporal and dispersion derivatives for the conditions:
  - `famous_1`,
 - `famous_2`,
 - `unfamiliar_1`,
 - `unfamiliar_2`,
 .

 Nuisance covariates included:
```

(continues on next page)

(continued from previous page)

```
 - `trans_?`,
 - `rot_?`,

to account for residual motion artefacts,
 .



 ## References

This method section was automatically generated using bidspm
(v2.2.0; https://github.com/cpp-lln-lab/bidspm; DOI: https://doi.org/10.5281/zenodo.
↪3554331)
and octache (https://github.com/Remi-Gau/Octache).
```

5.5 Parametric modulation

Those are not yet fully implemented but there is an example of how to get started in the face repetition demo folder.

```
{
  "Name": "parametric modulation",
  "BIDSModelVersion": "1.0.0",
  "Description": "model for face repetition",
  "Input": {
    "task": [
      "facerepetition"
    ],
    "space": [
      "IXI549Space"
    ]
  },
  "Nodes": [
    {
      "Level": "Run",
      "Name": "parametric",
      "GroupBy": [
        "run",
        "subject"
      ],
      "Transformations": {
        "Description": "merge the familiarity and repetition column to create the trial␣
↪type column",
        "Transformer": "bidspm",
        "Instructions": [
          {
            "Name": "Concatenate",
            "Input": [
              "face_type",
              "repetition_type"
```

(continues on next page)

(continued from previous page)

```
            ],
            "Output": "trial_type"
          }
        ]
      },
      "Model": {
        "X": [
          "trial_type.famous_first_show",
          "trial_type.famous_delayed_repeat",
          "trial_type.unfamiliar_first_show",
          "trial_type.unfamiliar_delayed_repeat",
          "trans_?",
          "rot_?"
        ],
        "HRF": {
          "Variables": [
            "trial_type.famous_first_show",
            "trial_type.famous_delayed_repeat",
            "trial_type.unfamiliar_first_show",
            "trial_type.unfamiliar_delayed_repeat"
          ],
          "Model": "spm"
        },
        "Type": "glm",
        "Options": {
          "HighPassFilterCutoffHz": 0.0078,
          "Mask": {
            "suffix": [
              "mask"
            ],
            "desc": [
              "brain"
            ]
          }
        },
        "Software": {
          "SPM": {
            "SerialCorrelation": "AR(1)",
            "ParametricModulations": [
              {
                "Name": "lag mod",
                "Conditions": [
                  "trial_type.famous_delayed_repeat",
                  "trial_type.unfamiliar_delayed_repeat"
                ],
                "Values": [
                  "lag"
                ],
                "PolynomialExpansion": 2
              }
            ]
          }
```

(continues on next page)

(continued from previous page)

```
      }
    },
    "DummyContrasts": {
      "Test": "t",
      "Contrasts": [
        "trial_type.famous_first_show",
        "trial_type.famous_delayed_repeat",
        "trial_type.unfamiliar_first_show",
        "trial_type.unfamiliar_delayed_repeat"
      ]
    },
    "Contrasts": [
      {
        "Name": "faces_gt_baseline",
        "ConditionList": [
          "trial_type.famous_first_show",
          "trial_type.famous_delayed_repeat",
          "trial_type.unfamiliar_first_show",
          "trial_type.unfamiliar_delayed_repeat"
        ],
        "Weights": [
          1,
          1,
          1,
          1
        ],
        "Test": "t"
      },
      {
        "Name": "faces_lt_baseline",
        "ConditionList": [
          "trial_type.famous_first_show",
          "trial_type.famous_delayed_repeat",
          "trial_type.unfamiliar_first_show",
          "trial_type.unfamiliar_delayed_repeat"
        ],
        "Weights": [
          -1,
          -1,
          -1,
          -1
        ],
        "Test": "t"
      }
    ]
  }
 ]
}
```

See the help section of `convertOnsetTsvToMat` for more information.

5.6 Examples

There are several examples of models in the model zoo along with links to their datasets.

Several of the *demos* have their own model and you can find several "dummy" models (without corresponding data) used for testing in this folder.

An example of JSON file could look something like that:

```
{
  "Name": "vislocalizer",
  "BIDSModelVersion": "1.0.0",
  "Description": "contrasts for the visual localizer",
  "Input": {
    "task": [
      "vislocalizer"
    ],
    "space": [
      "IXI549Space"
    ]
  },
  "Nodes": [
    {
      "Level": "Run",
      "Name": "run_level",
      "GroupBy": [
        "run",
        "subject"
      ],
      "Model": {
        "Type": "glm",
        "X": [
          "trial_type.VisMot",
          "trial_type.VisStat",
          "trial_type.missing_condition",
          "trans_?",
          "rot_?"
        ],
        "HRF": {
          "Variables": [
            "trial_type.VisMot",
            "trial_type.VisStat"
          ],
          "Model": "spm+derivative"
        },
        "Options": {
          "HighPassFilterCutoffHz": 0.008,
          "Mask": {
            "desc": [
              "brain"
            ],
            "suffix": [
              "mask"
            ]
```

(continues on next page)

(continued from previous page)

```
        }
      },
      "Software": {
        "SPM": {
          "InclusiveMaskingThreshold": 0.8,
          "SerialCorrelation": "FAST"
        },
        "bidspm": {
          "Results": [
            {
              "name": [
                "VisMot_&_VisStat"
              ],
              "p": 0.001,
              "MC": "none"
            },
            {
              "name": [
                "VisStat",
                "VisMot"
              ],
              "k": 10
            }
          ]
        }
      }
    },
    "DummyContrasts": {
      "Test": "t",
      "Contrasts": [
        "trial_type.VisMot",
        "trial_type.VisStat"
      ]
    },
    "Contrasts": [
      {
        "Name": "VisMot_&_VisStat",
        "ConditionList": [
          "trial_type.VisMot",
          "trial_type.VisStat"
        ],
        "Weights": [
          1,
          1
        ],
        "Test": "t"
      },
      {
        "Name": "VisMot_&_VisStat_lt_baseline",
        "ConditionList": [
          "trial_type.VisMot",
          "trial_type.VisStat"
```

(continues on next page)

(continued from previous page)

```
        ],
        "Weights": [
          -1,
          -1
        ],
        "Test": "t"
      }
    ]
  },
  {
    "Level": "Subject",
    "Name": "subject_level",
    "GroupBy": [
      "contrast",
      "subject"
    ],
    "Model": {
      "Type": "glm",
      "X": [
        1
      ],
      "Options": {
        "Mask": {
          "desc": [
            "brain"
          ],
          "suffix": [
            "mask"
          ]
        }
      },
      "Software": {
        "bidspm": {
          "Results": [
            {
              "name": [
                "VisMot_&_VisStat"
              ],
              "p": 0.001,
              "MC": "FDR"
            }
          ]
        }
      }
    },
    "DummyContrasts": {
      "Test": "t"
    }
  },
  {
    "Level": "Dataset",
    "Name": "dataset_level",
```

(continues on next page)

(continued from previous page)

```
      "GroupBy": [
        "contrast"
      ],
      "Model": {
        "Type": "glm",
        "X": [
          1
        ],
        "Options": {
          "Mask": {
            "desc": [
              "brain"
            ],
            "suffix": [
              "mask"
            ]
          }
        },
        "Software": {
          "bidspm": {
            "Results": [
              {
                "name": [
                  "VisMot_&_VisStat_lt_baseline"
                ],
                "p": 0.001,
                "binary": true
              }
            ]
          }
        }
      },
      "DummyContrasts": {
        "Test": "t"
      }
    }
  ]
}
```

CHAPTER

SIX

DEMOS

```
demos/
    ├── face_repetition
    ├── lesion_detection
    ├── MoAE
    ├── openneuro
    ├── tSNR
    └── vismotion
```

The demos show show you different way to use bidspm.

6.1 MoAE

```
/demos/MoAE
    ├── models
    └── options
```

This "Mother of All Experiments" is based on the block design dataset of SPM.

In the `options` folder has several examples of how to encode the options of your analysis in a json file.

In the `models` shows the BIDS statistical model used to run the GLM of this demo.

moae_01_bids_app

MoAE demo

This script shows how to use the bidspm BIDS app

- **Download**
  - download the dataset from the FIL for the block design SPM tutorial
- **Preprocessing**
  - copies the necessary data from the raw to the derivative folder,
  - runs spatial preprocessing

    those are otherwise handled by the workflows:
  - `bidsCopyInputFolder.m`
  - `bidsSpatialPrepro.m`
- **Stats**

  This will run the subject level GLM and contrasts on it of the MoaE dataset

- GLM specification + estimation
- compute contrasts
- show results

that are otherwise handled by the workflows

- `bidsFFX.m`
- `bidsResults.m`

Note: Results might be a bit different from those in the SPM manual as some default options are slightly different in this pipeline (e.g use of FAST instead of AR(1), motion regressors added)

---

type *bidspm help* or *bidspm('action', 'help')* or see this page: https://bidspm.readthedocs.io/en/stable/bids_app_api.html for more information on what parameters are obligatory or optional

`moae_fmriprep`

This script will run the FFX and contrasts on it of the MoAE dataset using the fmriprep preprocessed data

If you want to get the preprocessed data and you have datalad on your computer you can run the following commands to get the necessary data:

```
datalad install --source git@gin.g-node.org:/SPM_datasets/spm_moae_fmriprep.git \
        inputs/fmriprep
cd inputs/fmriprep && datalad get *.json \
                  */*/*tsv \
                  */*/*json \
                  */*/*desc-preproc*.nii.gz \
                  */*/*desc-brain*.nii.gz
```

Otherwise you also grab the data from OSF: https://osf.io/vufjs/download

`moae_02_create_roi_extract_data`

This script shows how to create a ROI and extract data from it.

Warning: This is "double dipping" as we use the same data to create the ROI we are going to extract the value from.

`moae_03_slice_display`

This script shows how to display the results of a GLM by having on the same image:

- the beta estimates
- the t statistics
- ROI contours

6.2 Face repetition

This is based on the event related design dataset of SPM.

`face_rep_01_bids_app`

This script will download the face repetition dataset from SPM and will run the basic preprocessing.

Download

- downloads and BIDSify the dataset from the FIL website

Preprocessing

- copies the necessary data from the raw to the derivative folder,
- runs slice time correction
- runs spatial preprocessing

those are otherwise handled by the workflows:

- `bidsCopyInputFolder.m`
- `bidsSTC.m`
- `bidsSpatialPrepro.m`

type *bidspm help* or *bidspm('action', 'help')* or see this page: https://bidspm.readthedocs.io/en/stable/bids_app_api.html for more information on what parameters are obligatory or optional

`face_rep_01_anat_only`

This show how an anat only pipeline would look like.

Download

- downloads and BIDSify the dataset from the FIL website

Preprocessing

- copies the necessary data from the raw to the derivative folder,
- runs spatial preprocessing

those are otherwise handled by the workflows:

- `bidsCopyInputFolder.m`
- `bidsSpatialPrepro.m`

type *bidspm help* or *bidspm('action', 'help')* or see this page: https://bidspm.readthedocs.io/en/stable/bids_app_api.html for more information on what parameters are obligatory or optional

`face_rep_02_stats`

> **Warning:** This script assumes you have already preprocessed the data with face_rep_01_bids_app.m

stats

This script will run the FFX and contrasts on the the face repetition dataset from SPM.

- GLM specification + estimation
- compute contrasts

- show results

that are otherwise handled by the workflows

- `bidsFFX.m`
- `bidsResults.m`

Note: Results might be a bit different from those in the SPM manual as some default options are slightly different in this pipeline (e.g use of FAST instead of AR(1), motion regressors added)

---

type *bidspm help* or *bidspm('action', 'help')* or see this page: https://bidspm.readthedocs.io/en/stable/bids_app_api.html for more information on what parameters are obligatory or optional

`face_rep_03_roi_analysis`

Creates a ROI in MNI space from the retinotopic probabilistic atlas.

Creates its equivalent in subject space (inverse normalization).

Then uses marsbar to run a ROI based GLM

6.3 Visual motion localizers

Small demo using visual motion localizer data to show how to set up an analysis with bidspm from scratch with datalad.

6.3.1 Using bidspm and datalad

Ideally better to use the datalad fMRI template we have set up, this shows a set by step approach.

Note: The bash script `vismotion_demo.sh` will run all the steps described here in one fell swoop.

You can run it by typing the following from within the bidspm/demos/vismotion

```
bash vismotion_demo.sh
```

Set up

Create a new datalad dataset with a YODA config

```
datalad create -c yoda visual_motion_localiser
cd visual_motion_localiser
```

Add the bidspm code as a sub-dataset, checkout the dev branch ands initializes

all submodules.

```
datalad install \
    -d . \
    -s https://github.com/cpp-lln-lab/bidspm.git \
    -r \
    code/bidspm
```

In case you get some errors when installing the submodules you might have to initialize them manually, and update your dataset with that update

```
cd code/bidspm
git checkout main
git submodule update --init --recursive && git submodule update --recursive
cd ..
datalad save -m 'update bidspm submodules'
```

Now let's get the raw data as a subdataset and put it in an `inputs/raw` folder.

The data from the CPP lab is openly available on GIN: https://gin.g-node.org/cpp-lln-lab/Toronto_VisMotionLocalizer_MR_raw

Note that to install it you will need to have set up Datalad to play nice with GIN: see the datalad handbook

This will install the data:

```
datalad install -d . \
                -s git@gin.g-node.org:/cpp-lln-lab/Trento_VisMotionLocalizer_MR_raw.git \
                --recursive \
                --jobs 12 \
                inputs/raw
```

After this your datalad dataset should look something like this:

```
├── code
│   └── bidspm
└── inputs
    └── raw
        ├── derivatives
        │   └── fmriprep
        ├── sub-con07
        ├── sub-con08
        └── sub-con15
```

To finish the setup you need to download the data:

```
cd inputs/raw
datalad get .
```

Note that you could have installed the dataset and got the data in one command:

Running the analysis

Start matlab and run the `step_1_preprocess.m` and `step_2_stats.m` scripts.

In the end your whole analysis should look like this.

```
├── code
│   └── bidspm
│       ├── binder
│       ├── demos
│       │   ├── face_repetition
│       │   ├── lesion_detection
│       │   ├── MoAE
│       │   ├── openneuro
│       │   ├── tSNR
│       │   └── vismotion          # <--- your scripts are there
│       ├── docs
│       ├── lib
│       ├── manualTests
│       ├── notebooks
│       ├── src
│       ├── templates
│       └── tests
├── inputs
│   └── raw                        # <--- input data
│       ├── derivatives
│       │   └── fmriprep           # <--- fmriprep data
│       ├── sub-con07
│       │   └── ses-01
│       ├── sub-con08
│       │   └── ses-01
│       └── sub-con15
│           └── ses-01
└── outputs
    └── derivatives
        ├── bidspm-preproc         # <--- smoothed data
        │   ├── jobs
        │   ├── sub-con07
        │   ├── sub-con08
        │   └── sub-con15
        └── bidspm-stats           # <--- stats output
            ├── jobs
            ├── sub-con07
            ├── sub-con08
            └── sub-con15
```

6.4 Openneuro based demos

6.4.1 Demos based on openneuro datasets

- ds000001: one task, one session, several runs
- ds000114: several tasks, several sessions, one or several runs depending on task
- ds001168: resting state, several sessions, several acquisition, fieldmaps, physio data
- ds002799: resting state and task, several sessions, with fmriprep data

Download with datalad

All those data can be installed with datalad.

Datalad datasets can be accessed via their siblings on: https://github.com/OpenNeuroDatasets

Check the content of the `Makefile` to see the code snippets you need to run to install those datasets.

Otherwise you can also get them by using the Datalad superdataset.

For example:

```
datalad install ///
cd datasets.datalad.org/
datalad install openneuro
datalad install openneuro/dsXXXXXX
cd openneuro/dsXXXXXX
# get rest data first subject
datalad get /openneuro/dsXXXXXX/sub-0001/func/sub-0001*
```

NARPS: ds001734

More details here:

https://docs.google.com/spreadsheets/d/1FU_F6kdxOD4PRQDIHXGHS4zTi_jEVaUqY_Zwg0z6S64/edit#gid=1019165812&range=A

TODO: add expected value to the model

```
% compute euclidean distance to the indifference line defined by
% gain twice as big as losses
% https://en.wikipedia.org/wiki/Distance_from_a_point_to_a_line
a = 0.5;
b = -1;
c = 0;
x = onsets{iRun}.gain;
y = onsets{iRun}.loss;
dist = abs(a * x + b * y + c) / (a^2 + b^2)^.5;
onsets{iRun}.EV = dist; % create an "expected value" regressor
```

TODO: transformers cannot yet be appled to confounds

```
{
    "Description": "Time points with a framewise displacement (as calculated by␣
↪fMRIprep) > 0.5 mm were censored (no interpolation) at the subject level GLM..",
    "Name": "Threshold",
    "Input": [
        "framewise_displacement"
    ],
    "Threshold": 0.5,
    "Binarize": true,
    "Output": [
        "thres_framewise_displacement"
    ]
},
```

CHAPTER

SEVEN

STATISTICS

Make sure you are familiar with the *BIDS stats model*, before you embark on to statistical analysis.

7.1 Statistics workflows

Note: The illustrations in this section mix what the files created by each workflow and the functions and are called by it. In this sense they are not pure DAGs (directed acyclic graphs) as the `*.m` files mentioned in them already exist.

7.1.1 Subject level

bidsFFX()

After the specification step an output folder is created. To get the fullpath of that folder you can use:

```
getFFXdir(subLabel, opt)
```

A typical folder will contain:

```
bidspm-stats/sub-01/stats/task-audio_space-IXI549Space_FWHM-6
├── SPM.mat
├── sub-01_task-audio_space-IXI549Space_desc-beforeEstimation_designmatrix.png
├── sub-01_task-audio_run-01_desc-confounds_regressors.mat
├── sub-01_task-audio_run-01_desc-confounds_regressors.tsv
├── sub-01_task-audio_run-01_onsets.mat
└── sub-01_task-audio_run-01_onsets.tsv
```

Each run should have a pair of tsv/mat files:

- One that summarises the onsets used for that design.
- One that summarises the regressors confounds used for that design.

In most cases those are going to be a subset of the content:

- of the `_events.tsv` from the raw BIDS dataset
- of the `_regressors.tsv` from the deriratives BIDS dataset containing the preprocessed data.

What part of the _events.tsv and _regressors.tsv gets into the final GLM specification depends on the BIDS statistical model used.

The mat files can directly be ingested by SPM: the TSV files are there for both logging and interoperability.

Fig. 1: Subject level GLM specification workflow for model specification

7.1.2 Compute results

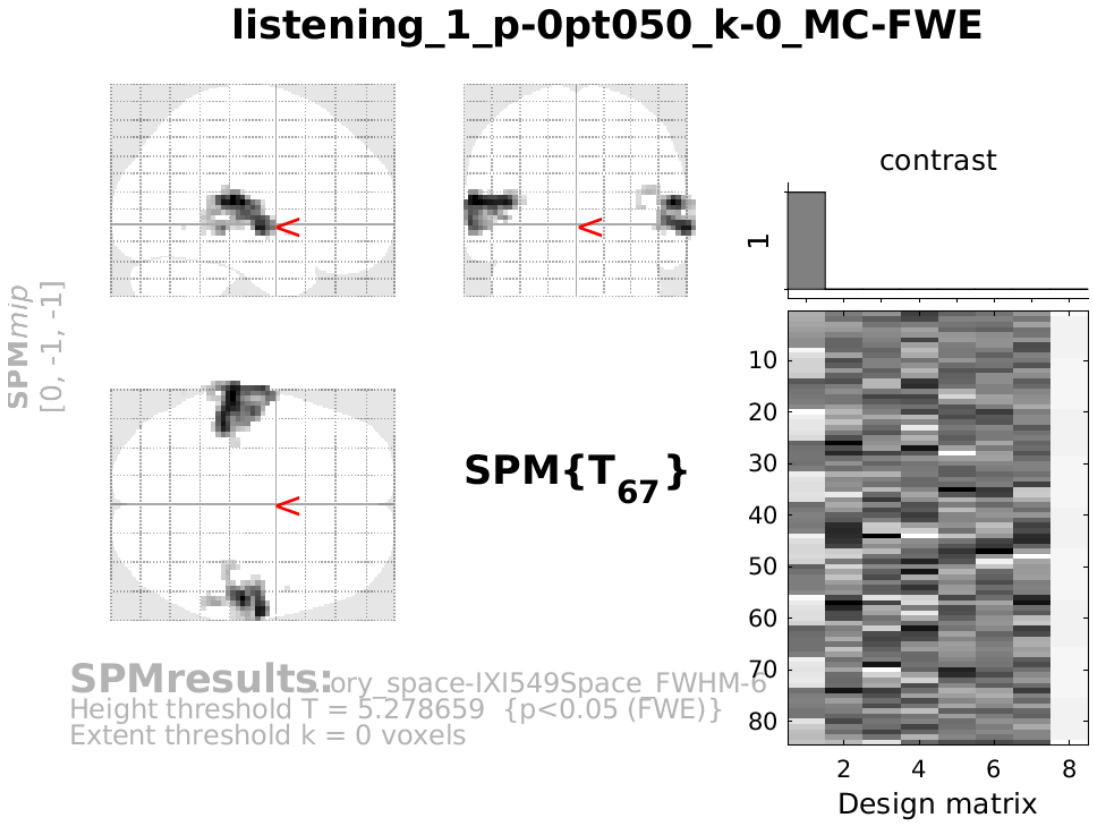
bidsResults()

`CSV output example`

bidspm also includes the `slice_display` code that allows you to plot on the same figure:

- beta values
- t values
- cluster boundaries
- ROI boundaries

An example of how to use it is available in the `moae_04_slice_display.m` script in the MoAE demo.



Statistics: ***p-values adjusted for search volume***

set-level		cluster-level				peak-level					mm	mm	mm
p	c	$p_{FWE-corr}$	$q_{FDR-corr}$	k_E	p_{uncorr}	$p_{FWE-corr}$	$q_{FDR-corr}$	T	(Z_E)	p_{uncorr}			
0.000	5	0.000	0.000	399	0.000	0.000	0.000	12.16	Inf	0.000	-63	-28	11
						0.000	0.000	11.33	Inf	0.000	-45	-34	11
						0.000	0.000	10.12	7.84	0.000	-66	-10	-1
		0.000	0.000	214	0.000	0.000	0.000	12.10	Inf	0.000	57	-22	11
						0.000	0.000	11.29	Inf	0.000	66	-13	-4
						0.000	0.003	7.07	6.09	0.000	60	-37	5
		0.001	0.020	5	0.012	0.006	0.200	5.81	5.21	0.000	69	-25	-4
		0.015	0.221	1	0.221	0.039	0.851	5.35	4.86	0.000	51	5	-7
		0.015	0.221	1	0.221	0.050	0.998	5.28	4.81	0.000	-63	-58	-4

table shows 3 local maxima more than 8.0mm apart

Height threshold: T = 5.28, p = 0.000 (0.050) Degrees of freedom = [1.0, 67.0]
Extent threshold: k = 0 voxels FWHM = 9.9 9.9 8.3 mm mm mm; 3.3 3.3 2.8 {voxels
Expected voxels per cluster, <k> = 0.715 Volume: 1784484 = 66092 voxels = 1953.4 resels
Expected number of clusters, <c> = 0.07 Voxel size: 3.0 3.0 3.0 mm mm mm; (resel = 30.35 vo
FWEp: 5.279, FDRp: 6.440, FWEc: 1, FDRc: 5

Fig. 2: Example of subject level results from the MoAE demo

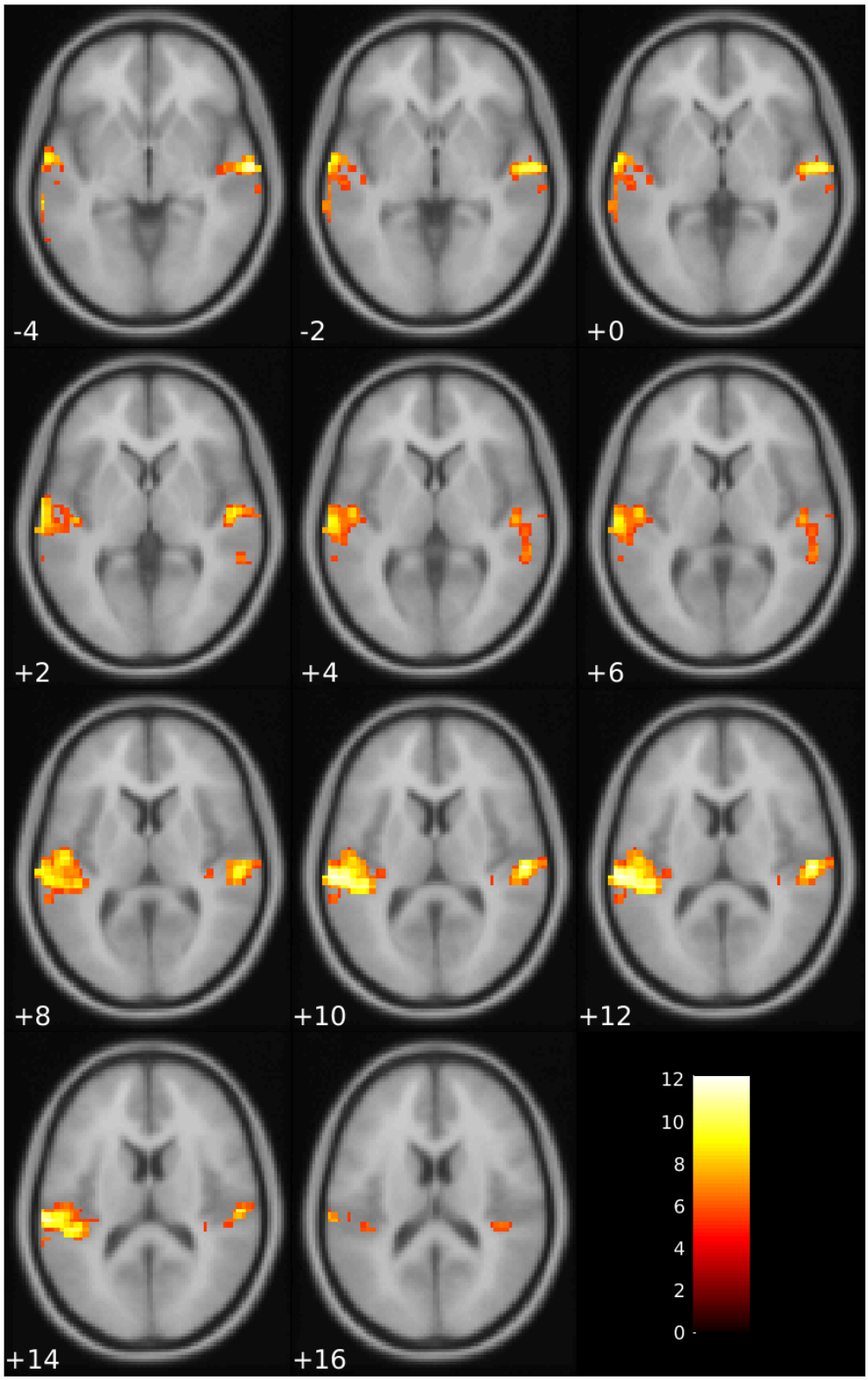


Fig. 3: Example of subject level montage from the MoAE demo

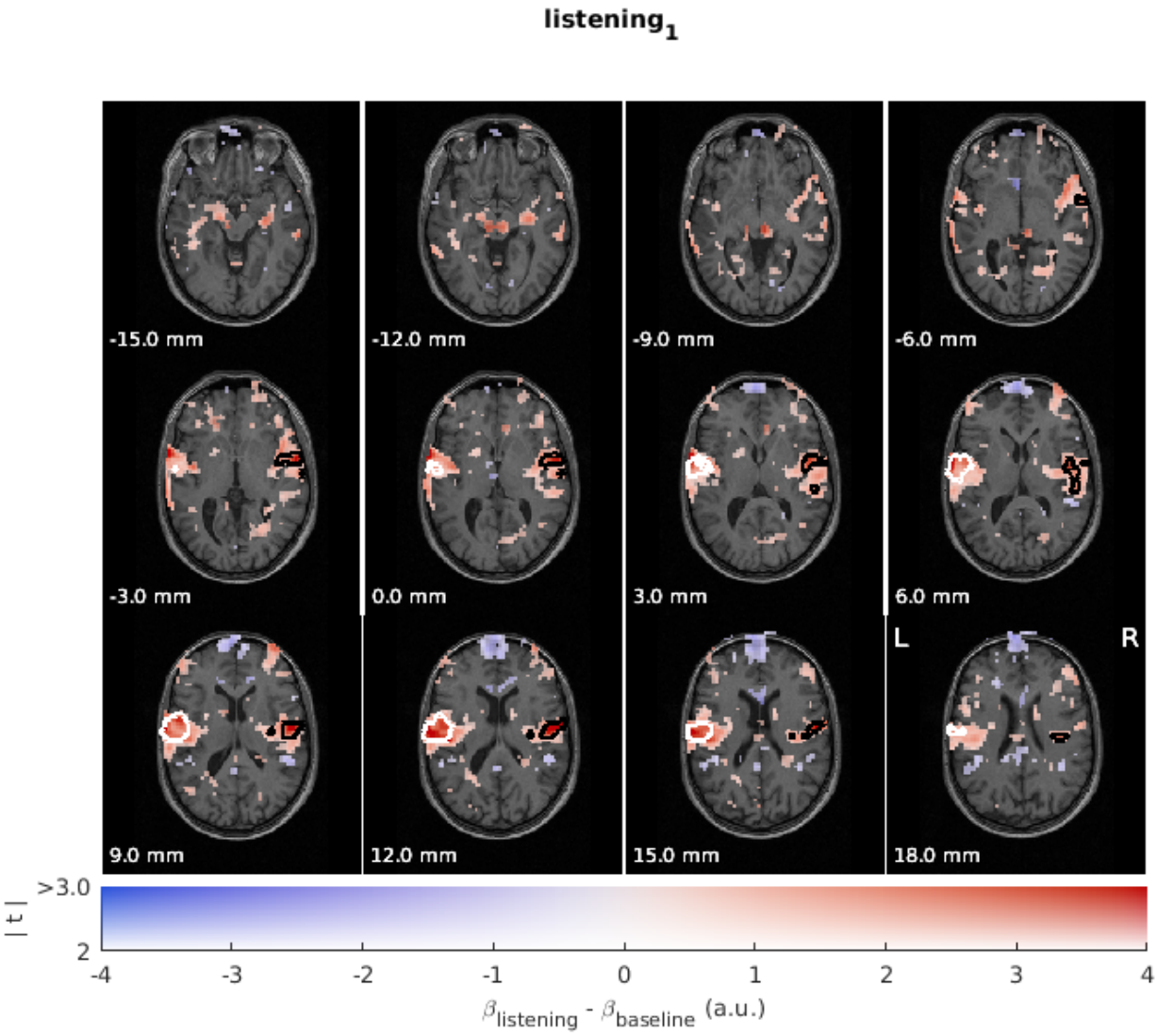


Fig. 4: Example of subject level slice display from the MoAE demo

CHAPTER

EIGHT

PREPROCESSING

8.1 Preprocessing workflows

Note: The illustrations in this section mix what the files created by each workflow and the functions and are called by it. In this sense they are not pure DAGs (directed acyclic graphs) as the `*.m` files mentioned in them already exist.

8.1.1 Slice Time Correction

bidsSTC()

More info available on this page of the SPM wikibook.

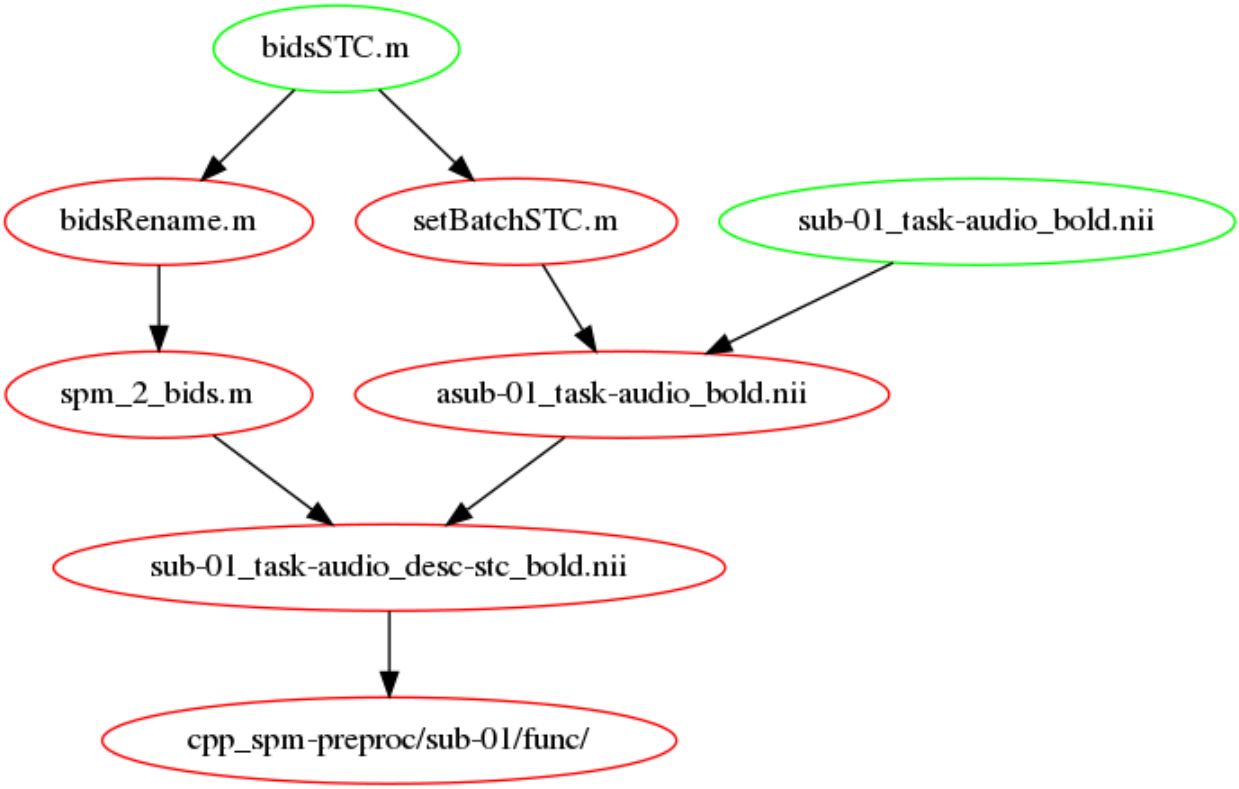


Fig. 1: Slice timing correction workflow

Some comments from here on STC, when it should be applied

At what point in the processing stream should you use it?

This is the great open question about slice timing, and it's not super-answerable. Both SPM and AFNI recommend you do it before doing realignment/motion correction, but it's not entirely clear why. The issue is this:

If you do slice timing correction before realignment, you might look down your non-realigned time course for a given voxel on the border of gray matter and CSF, say, and see one TR where the head moved and the voxel sampled from CSF instead of gray. This would results in an interpolation error for that voxel, as it would attempt to interpolate part of that big giant signal into the previous voxel. On the other hand, if you do realignment before slice timing correction, you might shift a voxel or a set of voxels onto a different slice, and then you'd apply the wrong amount of slice timing correction to them when you corrected - you'd be shifting the signal as if it had come from slice 20, say, when it actually came from slice 19, and shouldn't be shifted as much.

There's no way to avoid all the error (short of doing a four-dimensional realignment process combining spatial and temporal correction - Remi's note: fMRIprep does it), but I believe the current thinking is that doing slice timing first minimizes your possible error. The set of voxels subject to such an interpolation error is small, and the interpolation into another TR will also be small and will only affect a few TRs in the time course. By contrast, if one realigns first, many voxels in a slice could be affected at once, and their whole time courses will be affected. I think that's why it makes sense to do slice timing first. That said, here's some articles from the SPM e-mail list that comment helpfully on this subject both ways, and there are even more if you do a search for "slice timing AND before" in the archives of the list.

8.1.2 Spatial Preprocessing

Perform spatial preprocessing by running `bidsSpatialPrepro()`

The figures below show the `bidsSpatialPrepro` workflow as it would run using realign and unwarp (default) and with normalization to SPM MNI space (`IXI549Space`).

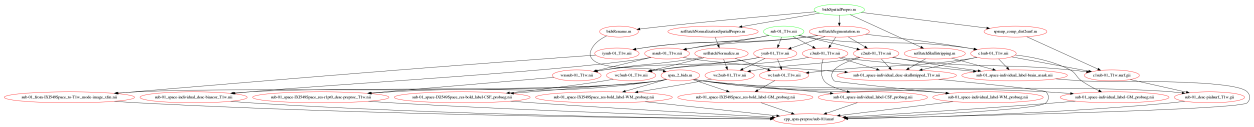


Fig. 2: Anatomical component of the spatial preprocessing workflow

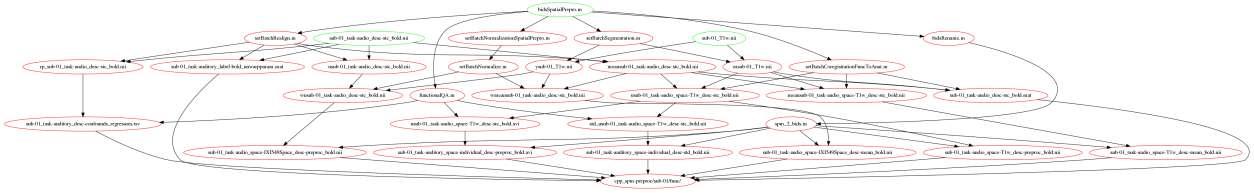


Fig. 3: Functional component of the spatial preprocessing workflow

8.1.3 Smoothing

Perform smoothing of the functional data by running *bidsSmoothing()*

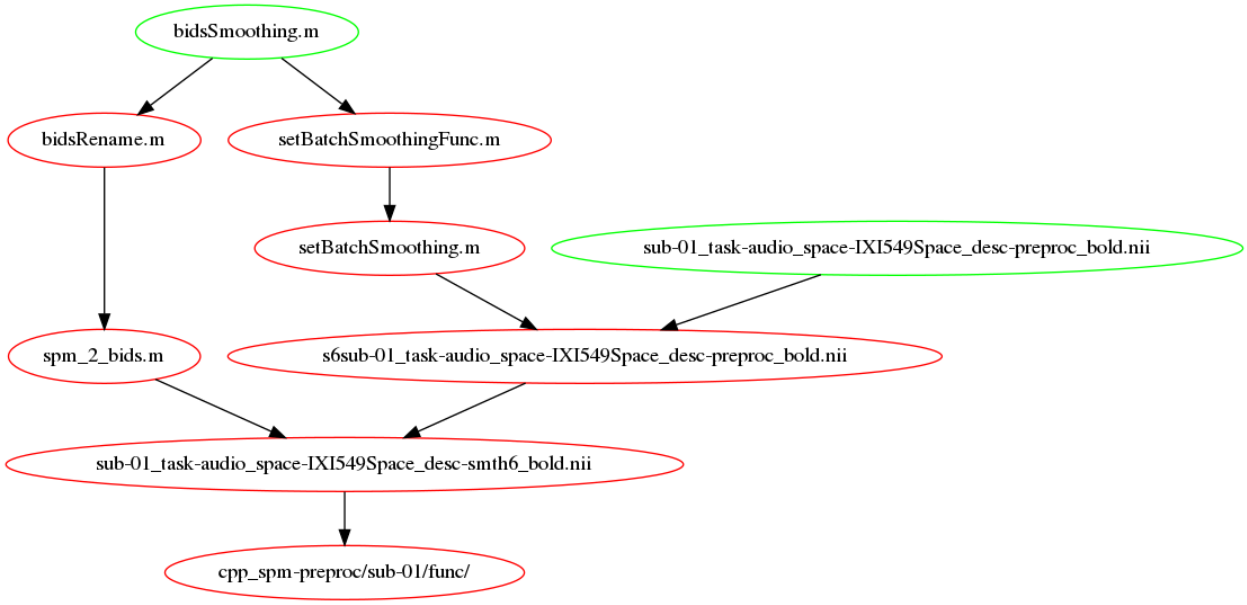


Fig. 4: Smoothing workflow

8.1.4 Others

bidsSegmentSkullStrip()

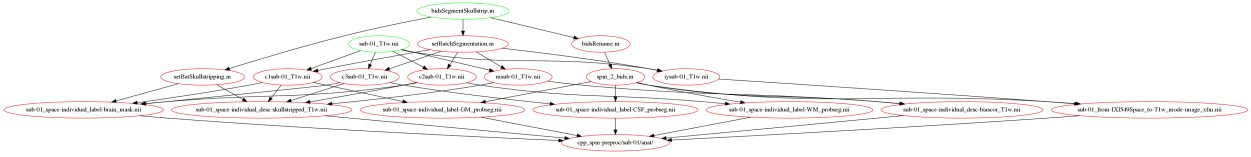


Fig. 5: Segment and skullstrip workflow

CHAPTER

NINE

OUTPUTS OF BIDSPM

9.1 jobs and logs

Each batch is saved in `jobs` folder in time a stamped m-file: those are more human readable and interoperable than their equivalent `.mat` file, and they can still be loaded in the SPM batch interface.

It is accompanied by a `.json` file that contains information about the environment in which the batch was run (operating system, bidspm version…).

```
bidspm-stats
├── jobs
│   └── taskName
│       ├── sub-01
│       │   ├── batch_batchName_task-taskName_space-space_FWHM-0_YYYY-MM-HHTMM-SS.json
│       │   └── batch_batchName_task_taskName_space_space_FWHM_0_YYYY-MM-HHTMM-SS.m
│       ├── sub-02
```

9.2 preprocessing

For a complete list of how SPM outputs are renamed into BIDS derivatives see the *Mapping* page.

9.2.1 func

SPM	BIDS
s6usub-01_task-auditory_bold.nii	sub-01_task-auditory_space-individual_desc-smth6_bold.nii
s6rsub-01_task-auditory_bold.nii	sub-01_task-auditory_space-individual_desc-smth6_bold.nii
s6uasub-01_task-auditory_bold.nii	sub-01_task-auditory_space-individual_desc-smth6_bold.nii
s6rasub-01_task-auditory_bold.nii	sub-01_task-auditory_space-individual_desc-smth6_bold.nii
s6wusub-01_task-auditory_bold.nii	sub-01_task-auditory_space-IXI549Space_desc-smth6_bold.nii
s6wrsub-01_task-auditory_bold.nii	sub-01_task-auditory_space-IXI549Space_desc-smth6_bold.nii
s6wuasub-01_task-auditory_bold.nii	sub-01_task-auditory_space-IXI549Space_desc-smth6_bold.nii
s6wrasub-01_task-auditory_bold.nii	sub-01_task-auditory_space-IXI549Space_desc-smth6_bold.nii
s6sub-01_task-auditory_bold.nii	sub-01_task-auditory_desc-smth6_bold.nii
rp_sub-01_task-auditory_bold.txt	sub-01_task-auditory_desc-confounds_regressors.tsv
rp_asub-01_task-auditory_bold.txt	sub-01_task-auditory_desc-confounds_regressors.tsv
usub-01_task-auditory_bold.nii	sub-01_task-auditory_space-individual_desc-preproc_bold.nii
uasub-01_task-auditory_bold.nii	sub-01_task-auditory_space-individual_desc-preproc_bold.nii
rsub-01_task-auditory_bold.nii	sub-01_task-auditory_space-individual_desc-preproc_bold.nii
rasub-01_task-auditory_bold.nii	sub-01_task-auditory_space-individual_desc-preproc_bold.nii
std_usub-01_task-auditory_bold.nii	sub-01_task-auditory_space-individual_desc-std_bold.nii
std_uasub-01_task-auditory_bold.nii	sub-01_task-auditory_space-individual_desc-std_bold.nii

9.2.2 anat

Note: Not listed:

- some of the outputs of the segmentation done by the ALI toolbox for lesion detection

SPM	BIDS
wc1sub-01_T1w.nii	sub-01_space-IXI549Space_res-bold_label-GM_probseg.nii
wc2sub-01_T1w.nii	sub-01_space-IXI549Space_res-bold_label-WM_probseg.nii
wc3sub-01_T1w.nii	sub-01_space-IXI549Space_res-bold_label-CSF_probseg.nii
rc1sub-01_T1w.nii	sub-01_space-individual_res-bold_label-GM_probseg.nii
rc2sub-01_T1w.nii	sub-01_space-individual_res-bold_label-WM_probseg.nii
rc3sub-01_T1w.nii	sub-01_space-individual_res-bold_label-CSF_probseg.nii
wmsub-01_T1w.nii	sub-01_space-IXI549Space_res-r1pt0_T1w.nii
wmsub-01_desc-skullstripped_T1w.nii	sub-01_space-IXI549Space_res-r1pt0_desc-preproc_T1w.nii
msub-01_desc-skullstripped_T1w.nii	sub-01_space-individual_desc-preproc_T1w.nii
wsub-01_desc-skullstripped_T1w.nii	sub-01_space-individual_res-r1pt0_desc-preproc_T1w.nii
msub-01_space-individual_desc-something_label-brain_mask.nii	sub-01_space-individual_label-brain_mask.nii
c1sub-01_T1w.surf.gii	sub-01_desc-pialsurf_T1w.gii

9.3 Statistics

At the subject level each folder contains for each run modeled:

- a pair of `*_onsets.mat` / `*_onsets.tsv`

The `*_onsets.mat` file contains the `names`, `onsets`, `durations`, `pmod` required by SPM to build the "multi condition" section of the model specification. The `*_onsets.tsv` is a human readable equivalent organised like BIDS `events.tsv` files.

- a pair of `*_desc-confounds_regressors.mat` / `*_desc-confounds_regressors.tsv`

The `*_desc-confounds_regressors.mat` file contains the `names`, `R` required by SPM to build the "multi regressor" section of the model specification. The `*_desc-confounds_regressors.tsv` is a human readable equivalent organised like BIDS derivatives `timeseries.tsv` files.

```
bidspm-stats
├── sub-01
│   └── task-taskName_space-space_FWHM-0_node-nodeName
│       |
│       ├── beta_0001.nii  ----------
│       ├── beta_*.nii              |
│       ├── con_0001.nii            |
│       ├── con_*.nii               |
│       ├── mask.nii                | Regular SPM output
│       ├── ResMS.nii               |
│       ├── RPV.nii                 |
│       ├── SPM.mat                 |
│       ├── spmT_0001.nii           |
│       ├── spmT_*.nii     ----------
│       |
│       ├── sub-blnd01_task-taskName_space-space_desc-contrastName_label-0039_p-0pt050_k-
↪10_MC-FWE_montage.png
│       |
│       ├── sub-blnd01_task-taskName_space-space_desc-afterEstimation_designmatrix.png
│       ├── sub-blnd01_task-taskName_space-space_desc-beforeEstimation_designmatrix.png
│       |
│       ├── sub-blnd01_task-taskName_run-01_desc-confounds_regressors.mat ------
│       ├── sub-blnd01_task-taskName_run-01_desc-confounds_regressors.tsv      |
│       ├── sub-blnd01_task-taskName_run-01_onsets.mat                         |
│       ├── sub-blnd01_task-taskName_run-01_onsets.tsv                         | Files␣
↪used for model specification
│       ├── sub-blnd01_task-taskName_run-*_desc-confounds_regressors.mat       |
│       ├── sub-blnd01_task-taskName_run-*_desc-confounds_regressors.tsv       |
│       ├── sub-blnd01_task-taskName_run-*_onsets.mat                         |
│       └── sub-blnd01_task-taskName_run-*_onsets.tsv --------------------------
...
```

```
outputs/derivatives/bidspm-stats
├── CHANGES
├── dataset_description.json
├── derivatives
│   └── bidspm-groupStats
│       ├── CHANGES
```

(continues on next page)

(continued from previous page)

```
│       ├── dataset_description.json
│       ├── jobs
│       │   └── <taskLabel>
│       │       └── group
│       │           ├── batch_contrasts_rfx_2023-05-06T12-49.json
│       │           ├── batch_contrasts_rfx_2023_05_06T12_49.m
│       │           ├── batch_group_level_model_specification_estimation_2023-05-06T12-
↪49.json
│       │           └── batch_group_level_model_specification_estimation_2023_05_06T12_
↪49.m
│       ├── LICENSE
│       ├── README.md
│       ├── sub-ALL_task-<taskLabel>_space-<spaceLabel>_FWHM-6_conFWHM-0_node-dataset_
↪contrast-<taskLabel>
│       │   ├── beta_0001.nii
│       │   ├── con_0001.nii
│       │   ├── mask.nii
│       │   ├── ResMS.nii
│       │   ├── RPV.nii
│       │   ├── SPM.mat
│       │   ├── spmT_0001.nii
│       │   ├── task-<taskLabel>_space-<spaceLabel>_desc-afterEstimation_designmatrix.png
│       │   └── task-<taskLabel>_space-<spaceLabel>_desc-beforeEstimation_designmatrix.
↪png
│       └── sub-ALL_task-<taskLabel>_space-<spaceLabel>_FWHM-6_conFWHM-6_node-dataset_
↪contrast-<taskLabel>
│           ├── beta_0001.nii
│           ├── con_0001.nii
│           ├── mask.nii
│           ├── ResMS.nii
│           ├── RPV.nii
│           ├── SPM.mat
│           ├── spmT_0001.nii
│           ├── task-<taskLabel>_space-<spaceLabel>_desc-afterEstimation_designmatrix.png
│           └── task-<taskLabel>_space-<spaceLabel>_desc-beforeEstimation_designmatrix.
↪png
├── jobs
│   └── <taskLabel>
│       ├── group
│       │   ├── batch_smooth_con_FWHM-6_task-<taskLabel>_2023-05-06T12-49.json
│       │   └── batch_smooth_con_FWHM_6_task_<taskLabel>_2023_05_06T12_49.m
│       ├── sub-292
│       │   ├── batch_contrasts_ffx_task-<taskLabel>_space-<spaceLabel>_FWHM-6_2023-05-
↪06T12-43.json
│       │   ├── batch_contrasts_ffx_task_<taskLabel>_space_<spaceLabel>_FWHM_6_2023_05_
↪06T12_43.m
│       │   ├── batch_specifyAndEstimate_ffx_task-<taskLabel>_space-<spaceLabel>_FWHM-6_
↪2023-05-06T12-35.json
│       │   └── batch_specifyAndEstimate_ffx_task_<taskLabel>_space_<spaceLabel>_FWHM_6_
↪2023_05_06T12_35.m
│       ├── sub-302
│       └── sub-307
```

(continues on next page)

(continued from previous page)

```
├── LICENSE
├── README.md
├── reports
│   ├── bidspm.bib
│   └── stats_model-default_es_model_citation.md
├── sub-292
│   └── task-<taskLabel>_space-<spaceLabel>_FWHM-6
│       ├── beta_0001.nii
│       ├── beta_0002.nii
│       ├── ...
│       ├── beta_0060.nii
│       ├── con_0001.nii
│       ├── con_0002.nii
│       ├── con_0003.nii
│       ├── con_0004.nii
│       ├── con_0005.nii
│       ├── con_0006.nii
│       ├── mask.nii
│       ├── ResMS.nii
│       ├── RPV.nii
│       ├── s6con_0001.nii
│       ├── s6con_0002.nii
│       ├── s6con_0003.nii
│       ├── s6con_0004.nii
│       ├── s6con_0005.nii
│       ├── s6con_0006.nii
│       ├── SPM.mat
│       ├── spmT_0001.nii
│       ├── spmT_0002.nii
│       ├── spmT_0003.nii
│       ├── spmT_0004.nii
│       ├── spmT_0005.nii
│       ├── spmT_0006.nii
│       ├── sub-292_ses-<sesLabel>_task-<taskLabel>_run-01_desc-confounds_regressors.mat
│       ├── sub-292_ses-<sesLabel>_task-<taskLabel>_run-01_desc-confounds_regressors.tsv
│       ├── sub-292_ses-<sesLabel>_task-<taskLabel>_run-01_onsets.mat
│       ├── sub-292_ses-<sesLabel>_task-<taskLabel>_run-01_onsets.tsv
│       ├── sub-292_task-<taskLabel>_space-<spaceLabel>_desc-afterEstimation_
↪designmatrix.png
│       └── sub-292_task-<taskLabel>_space-<spaceLabel>_desc-beforeEstimation_
↪designmatrix.png
├── sub-302
│   └── task-<taskLabel>_space-<spaceLabel>_FWHM-6
│       ├── beta_0001.nii
│       ├── beta_0002.nii
│       ├── ...
│       ├── beta_0154.nii
│       ├── con_0001.nii
│       ├── con_0002.nii
│       ├── con_0003.nii
│       ├── mask.nii
```

(continues on next page)

(continued from previous page)

```
│       ├── ResMS.nii
│       ├── RPV.nii
│       ├── s6con_0001.nii
│       ├── s6con_0002.nii
│       ├── s6con_0003.nii
│       ├── SPM.mat
│       ├── spmT_0001.nii
│       ├── spmT_0002.nii
│       ├── spmT_0003.nii
│       ├── sub-302_ses-<sesLabel>_task-<taskLabel>_run-01_desc-confounds_regressors.mat
│       ├── sub-302_ses-<sesLabel>_task-<taskLabel>_run-01_desc-confounds_regressors.tsv
│       ├── sub-302_ses-<sesLabel>_task-<taskLabel>_run-01_onsets.mat
│       ├── sub-302_ses-<sesLabel>_task-<taskLabel>_run-01_onsets.tsv
│       ├── sub-302_task-<taskLabel>_space-<spaceLabel>_desc-afterEstimation_
↪designmatrix.png
│       └── sub-302_task-<taskLabel>_space-<spaceLabel>_desc-beforeEstimation_
↪designmatrix.png
└── sub-307
    └── task-<taskLabel>_space-<spaceLabel>_FWHM-6
        ├── beta_0001.nii
        ├── beta_0002.nii
        ├── ...
        ├── beta_0237.nii
        ├── con_0001.nii
        ├── con_0002.nii
        ├── con_0003.nii
        ├── con_0004.nii
        ├── con_0005.nii
        ├── con_0006.nii
        ├── con_0007.nii
        ├── con_0008.nii
        ├── mask.nii
        ├── ResMS.nii
        ├── RPV.nii
        ├── s6con_0001.nii
        ├── s6con_0002.nii
        ├── s6con_0003.nii
        ├── s6con_0004.nii
        ├── s6con_0005.nii
        ├── s6con_0006.nii
        ├── s6con_0007.nii
        ├── s6con_0008.nii
        ├── SPM.mat
        ├── spmT_0001.nii
        ├── spmT_0002.nii
        ├── spmT_0003.nii
        ├── spmT_0004.nii
        ├── spmT_0005.nii
        ├── spmT_0006.nii
        ├── spmT_0007.nii
        ├── spmT_0008.nii
        ├── sub-307_ses-<sesLabel>_task-<taskLabel>_run-01_desc-confounds_regressors.mat
```

(continues on next page)

(continued from previous page)

```
        ├── sub-307_ses-<sesLabel>_task-<taskLabel>_run-01_desc-confounds_regressors.tsv
        ├── sub-307_ses-<sesLabel>_task-<taskLabel>_run-01_onsets.mat
        ├── sub-307_ses-<sesLabel>_task-<taskLabel>_run-01_onsets.tsv
        ├── sub-307_task-<taskLabel>_space-<spaceLabel>_desc-afterEstimation_
↪designmatrix.png
        └── sub-307_task-<taskLabel>_space-<spaceLabel>_desc-beforeEstimation_
↪designmatrix.png
```

CHAPTER

TEN

MAPPING

input	output
m	*space-individual_desc-biascor
c1	*space-individual_label-GM_probseg
c2	*space-individual_label-WM_probseg
c3	*space-individual_label-CSF_probseg
iy_	*from-IXI549Space_to-T1w_mode-image_xfm
y_	*from-T1w_to-IXI549Space_mode-image_xfm
segparam_	*segparam
a	*space-individual_desc-stc
au	*space-individual_desc-stc
unwarpparam_	*unwarpparam
u	*space-individual_desc-preproc
ua	*space-individual_desc-preproc
rp_	*motion.tsv
rp_a	*motion.tsv
rp_au	*motion.tsv
wc1	*space-IXI549Space_label-GM_res-bold_probseg
wc2	*space-IXI549Space_label-WM_res-bold_probseg
wc3	*space-IXI549Space_label-CSF_res-bold_probseg
s	*space-individual_desc-smth
sua	*space-individual_desc-smth
sau	*space-individual_desc-smth
sra	*space-individual_desc-smth
su	*space-individual_desc-smth
sr	*space-individual_desc-smth
sw	*space-IXI549Space_desc-smth
swua	*space-IXI549Space_desc-smth
swau	*space-IXI549Space_desc-smth
swra	*space-IXI549Space_desc-smth
swu	*space-IXI549Space_desc-smth
swr	*space-IXI549Space_desc-smth
w	*space-IXI549Space_desc-preproc
wm	*space-IXI549Space_res-r1pt0
wau	*space-IXI549Space_desc-preproc
wua	*space-IXI549Space_desc-preproc
wra	*space-IXI549Space_desc-preproc
wu	*space-IXI549Space_desc-preproc
wr	*space-IXI549Space_desc-preproc
ra	*space-individual_desc-preproc

continues on next page

Table 1 – continued from previous page

input	output
mean	*space-individual_desc-mean
meanu	*space-individual_desc-mean
meanua	*space-individual_desc-mean
meanau	*space-individual_desc-mean
wmean	*space-IXI549Space_desc-mean
wmeanu	*space-IXI549Space_desc-mean
wmeanua	*space-IXI549Space_desc-mean
wmeanau	*space-IXI549Space_desc-mean
rc1	*space-individual_label-GM_res-bold_probseg
rc2	*space-individual_label-WM_res-bold_probseg
rc3	*space-individual_label-CSF_res-bold_probseg
s6w	*space-IXI549Space_desc-smth6
s6wua	*space-IXI549Space_desc-smth6
s6wra	*space-IXI549Space_desc-smth6
s6wu	*space-IXI549Space_desc-smth6
s6wr	*space-IXI549Space_desc-smth6
s6ua	*space-individual_desc-smth6
s6ra	*space-individual_desc-smth6
s6u	*space-individual_desc-smth6
s6r	*space-individual_desc-smth6
s6	*desc-smth6
mean_ua	*space-individual_desc-mean
mean_r	*space-individual_desc-mean
mean_u	*space-individual_desc-mean
mean_	*space-individual_desc-mean
std_ua	*space-individual_desc-std
std_u	*space-individual_desc-std
std_r	*space-individual_desc-std
std_	*space-individual_desc-std
mdesc-skullstripped_T1w.nii	*space-individual_desc-preproc
mlabel-brain_mask.nii	*label-brain
c1add-joker_T1w.surf.gii	*desc-pialsurf.gii
wdesc-skullstripped_T1w.nii	*space-IXI549Space_desc-preproc_res-r1pt0
wdesc-brain_mask.nii	*space-IXI549Space_desc-brain_res-r1pt0
wmdesc-skullstripped_T1w.nii	*space-IXI549Space_desc-preproc_res-r1pt0

CHAPTER

ELEVEN

METHODS SECTION

11.1 Dataset description

Use the *bidsReport()* function to description of your dataset that can be used for your methods section

11.2 Preprocessing & GLM

This can be generated with the `boilerplate()` function.

11.2.1 Output example - Preprocessing

Pre processing

The (f)MRI data were pre-processed with bidspm (v2.2.0; https://github.com/cpp-lln-lab/bidspm; DOI: https://doi.org/10.5281/zenodo.3554331) using statistical parametric mapping (SPM12 - 7771; Wellcome Center for Neuroimaging, London, UK; https://www.fil.ion.ucl.ac.uk/spm; RRID:SCR_007037) using MATLAB 9.4.0.813654 (R2018a) on a unix computer (Linux version 5.15.0-53-generic (build@lcy02-amd64-047) (gcc (Ubuntu 11.2.0-19ubuntu1) 11.2.0, GNU ld (GNU Binutils for Ubuntu) 2.38) #59-Ubuntu SMP Mon Oct 17 18:53:30 UTC 2022). .

The preprocessing of the functional images was performed in the following order:

- removing of dummy scans
- slice timing correction
- realignment and unwarping
- segmentation and skullstripping
- normalization MNI space
- smoothing

{{nb}} dummy scans were removed to allow for signal stabilization.

Slice timing correction was performed taking the ^th slice as a reference (interpolation: sinc interpolation).

Functional scans from each participant were realigned and unwarped using the mean image as a reference (SPM single pass; number of degrees of freedom: 6 ; cost function: least square) (Friston et al, 1995).

The anatomical image was bias field corrected. The bias field corrected image was segmented and normalized to MNI space (target space: IXI549Space; target resolution: 1 mm; interpolation: 4th degree b-spline) using a unified segmentation.

The tissue probability maps generated by the segmentation were used to skullstripp the bias corrected image removing any voxel with p(gray matter) + p(white matter) + p(CSF) > 0.75.

The mean functional image obtained from realignement was co-registered to the bias corrected anatomical image (number of degrees of freedom: 6 ; cost function: normalized mutual information) (Friston et al, 1995). The transformation matrix from this coregistration was applied to all the functional images.

The deformation field obtained from the segmentation was applied to all the functional images (target space: IXI549Space; target resolution: equal to that used at acquisition; interpolation: 4th degree b-spline).

Preprocessed functional images were spatially smoothed using a 3D gaussian kernel (FWHM = 6 mm).

References

This method section was automatically generated using bidspm (v2.2.0; https://github.com/cpp-lln-lab/bidspm; DOI: https://doi.org/10.5281/zenodo.3554331) and octache (https://github.com/Remi-Gau/Octache).

11.2.2 Output example - GLM subject level

fMRI statistical analysis

The fMRI data were analysed with bidspm (v2.2.0; https://github.com/cpp-lln-lab/bidspm; DOI: https://doi.org/10.5281/zenodo.3554331) using statistical parametric mapping (SPM12 - 7771; Wellcome Center for Neuroimaging, London, UK; https://www.fil.ion.ucl.ac.uk/spm; RRID:SCR_007037) using MATLAB 9.4.0.813654 (R2018a) on a unix computer (Linux version 5.15.0-53-generic (build@lcy02-amd64-047) (gcc (Ubuntu 11.2.0-19ubuntu1) 11.2.0, GNU ld (GNU Binutils for Ubuntu) 2.38) #59-Ubuntu SMP Mon Oct 17 18:53:30 UTC 2022).

The input data were the preprocessed BOLD images in IXI549Space space for the task " facerepetition ".

Run / subject level analysis

At the subject level, we performed a mass univariate analysis with a linear regression at each voxel of the brain, using generalized least squares with a global AR(1) model to account for temporal auto-correlation and a drift fit with discrete cosine transform basis (128 seconds cut-off).

Image intensity scaling was done run-wide before statistical modeling such that the mean image would have a mean intracerebral intensity of 100.

We modeled the fMRI experiment in a event design with regressors entered into the run-specific design matrix. The onsets were convolved with SPM canonical hemodynamic response function (HRF) and its temporal and dispersion derivatives for the conditions:

- `famous_1`,
- `famous_2`,
- `unfamiliar_1`,
- `unfamiliar_2`, .

Nuisance covariates included:

- `trans_?`,
- `rot_?`,

to account for residual motion artefacts, .

References

This method section was automatically generated using bidspm (v2.2.0; https://github.com/cpp-lln-lab/bidspm; DOI: https://doi.org/10.5281/zenodo.3554331) and octache (https://github.com/Remi-Gau/Octache).

11.2.3 Output example - GLM Group level

CHAPTER

TWELVE

FREQUENTLY ASKED QUESTIONS

- *BIDS: What is a BIDS way to name and store Regions of Interest (ROIs)?*
- *General: Can I use parallelization with bidspm?*
- *General: How can I know that things are set up properly before I run an analysis?*
- *General: How can I prevent from having SPM windows pop up all the time?*
- *General: How can I run bidspm from the command line?*
- *General: How can I run bidspm only certain files, like just the session* `02` *for example?*
- *General: What happens if we run same code twice?*
- *Results: How can I change which slices are shown in a montage?*
- *SPM: How can get the data from some specific voxel in a nifti image?*
- *SPM: How can I use SPM image calculator (*`imcalc`*) to create a ROI?*
- *SPM: How can set the value of a specific voxel in a nifti image?*
- *SPM: How do I get the header of a nifti image?*
- *SPM: How do I know which matlab function performs a given SPM process?*
- *SPM: How do I merge 2 masks with SPM?*
- *SPM: What is the content of the SPM.mat file?*
- *Statistics: How can change the name of the folder of the subject level analysis?*
- *Statistics: How can I see what the transformation in the BIDS stats model will do to my events.tsv?*
- *Statistics: How should I name my conditions in my events.tsv?*
- *Statistics: How should I structure my data to run my statistical analysis?*

12.1 BIDS: What is a BIDS way to name and store Regions of Interest (ROIs)?

There is no "official" way to name ROI in BIDS, but you can apply BIDS naming principles to name those.

The closest things to ROI naming are the `masks` for the BIDS derivatives.

Here is an example from the :ref:`face repetition demo`::

```
bidspm-roi
├── group
│   ├── hemi-L_space-MNI_label-V1d_desc-wang_mask.json
│   ├── hemi-L_space-MNI_label-V1d_desc-wang_mask.nii
│   ├── hemi-L_space-MNI_label-V1v_desc-wang_mask.json
│   ├── hemi-L_space-MNI_label-V1v_desc-wang_mask.nii
│   ├── hemi-R_space-MNI_label-V1d_desc-wang_mask.json
│   ├── hemi-R_space-MNI_label-V1d_desc-wang_mask.nii
│   ├── hemi-R_space-MNI_label-V1v_desc-wang_mask.json
│   └── hemi-R_space-MNI_label-V1v_desc-wang_mask.nii
└── sub-01
    └── roi
        ├── sub-01_hemi-L_space-individual_label-V1d_desc-wang_mask.nii
        ├── sub-01_hemi-L_space-individual_label-V1v_desc-wang_mask.nii
        ├── sub-01_hemi-R_space-individual_label-V1d_desc-wang_mask.nii
        └── sub-01_hemi-R_space-individual_label-V1v_desc-wang_mask.nii
```

ROIs that are defined in some MNI space are going to be the same across subjects, so you could store a "group" folder (this is not BIDSy but is less redundant than having a copy of the same file for each subject).

The `desc` entity (description) here is used to denotate the atlas the ROI taken from, so if you are building yours from a localizer you might not need to use it.

Ideally you would want to add a JSON file to add metadata about those ROIs.

You can use bids-matlab to help you create BIDS valid filenames.

```
>> name_spec.ext = '.nii';
>> name_spec.suffix = 'mask';
>> name_spec.entities = struct( ...
                              'hemi', 'R', ...
                              'space', 'MNI', ...
                              'label', 'V1v', ...
                              'desc', 'wang');
>> file = bids.File(name_spec);
>> file.filename

   hemi-R_space-MNI_label-V1v_desc-wang_mask.nii
```

12.2 General: Can I use parallelization with bidspm?

You can use parallelization with bidspm. The main way to do so is to use `parfor` loops in your code when running the `preprocess`, `smooth` or `stats` actions.

For example, if you want to run the `preprocess` action on a list of subjects,

```
subjects = {'01', '02', '03', '04', '05'};
parfor i = 1:length(subjects)
    bidspm(bids_dir, output_dir, ...
           'participant_label', subjects(i), ...
           'action', 'preprocess', ...
           'task', {'tasnName'}, ...
           'space', {'IXI549Space'});
end
```

12.3 General: How can I know that things are set up properly before I run an analysis?

If you want to set things up but not let SPM actually run the batches you can use the option:

`opt.dryRun = true()`

This can be useful when debugging. You may still run into errors when SPM jobman takes over and starts running the batches, but you can at least see if the batches will be constructed without error and then inspect with the SPM GUI to make sure everything is fine.

12.4 General: How can I prevent from having SPM windows pop up all the time?

Running large number of batches when the GUI of MATLAB is active can be annoying, as SPM windows will always pop up and become active instead of running in the background like most users would prefer to.

One easy solution is to add a `spm_my_defaults` function with the following content in the MATLAB path, or in the directory where you are running your scripts or command from.

```
function spm_my_defaults

  global defaults

  defaults.cmdline = true;

end
```

This should be picked up by bidspm and SPM upon initialization and ensure that SPM runs in command line mode.

12.5 General: How can I run bidspm from the command line?

You can use the Python CLI of bidspm to run many functionalities from the terminal.

See the README to see how to install it.

You can also run your matlab script from within your terminal without starting the MATLAB graphic interface.

For this you first need to know where is the MATLAB application. Here are the typical location depending on your operating system (where `XXx` corresponds to the version you use).

- Windows: `C:\Program Files\MATLAB\R20XXx\bin\matlab.exe`
- Mac: `/Applications/Matlab_R20XXx.app/bin/matlab`
- Linux: `/usr/local/MATLAB/R20XXx/bin/matlab`

You can then launch MATLAB from a terminal in a command line only with the following arguments: `-nodisplay -nosplash -nodesktop`

So on Linux for example:

```
/usr/local/MATLAB/R2017a/bin/matlab -nodisplay -nosplash -nodesktop
```

If you are on Mac or Linux, we would recommend adding those aliases to your `.bashrc` or wherever else you keep your aliases.

```
matlab=/usr/local/MATLAB/R20XXx/bin/matlab
matlabcli='/usr/local/MATLAB/R20XXx/bin/matlab -nodisplay -nosplash -nodesktop'
```

12.6 General: How can I run bidspm only certain files, like just the session `02` for example?

Currently there are 2 ways of doing this.

- using the `bids_filter_file` parameter or an equivalent `bids_filter_file.json` file or its counterpart field `opt.bidsFilterFile`
- using the `opt.query` option field.

On the long run the plan is to use only the `bids_filter_file`, but for now both possibilities should work.

12.6.1 `bids filter file`

This is similar to the way you can "select" only certain files to preprocess with fmriprep.

You can use a `opt.bidsFilterFile` field in your options to define a typical images "bold", "T1w" in terms of their BIDS entities. The default value is:

```
struct('fmap', struct('modality', 'fmap'), ...
       'bold', struct('modality', 'func', 'suffix', 'bold'), ...
       't2w',  struct('modality', 'anat', 'suffix', 'T2w'), ...
       't1w',  struct('modality', 'anat', 'space', '', 'suffix', 'T1w'), ...
       'roi',  struct('modality', 'roi', 'suffix', 'mask'));
```

Similarly when using the bidspm you can use the argument `bids_filter_file` to pass a structure or point to a JSON file that would also define typical images "bold", "T1w"…

The default content in this case would be:

```
{
    "fmap": { "datatype": "fmap" },
    "bold": { "datatype": "func", "suffix": "bold" },
    "t2w": { "datatype": "anat", "suffix": "T2w" },
    "t1w": { "datatype": "anat", "space": "", "suffix": "T1w" },
    "roi": { "datatype": "roi", "suffix": "mask" }
}
```

So if you wanted to run your analysis on say run `02` and `05` of session `02`, you would define a json file this file like this:

```
{
    "fmap": { "datatype": "fmap" },
    "bold": {
        "datatype": "func",
        "suffix": "bold",
        "ses": "02",
        "run": ["02", "05"]
    },
    "t2w": { "datatype": "anat", "suffix": "T2w" },
    "t1w": { "datatype": "anat", "space": "", "suffix": "T1w" },
    "roi": { "datatype": "roi", "suffix": "mask" }
}
```

12.6.2 `opt.query`

You can select a subset of your data by using the `opt.query`.

This will create a "filter" that bids-matlab will use to only "query" and retrieve the subset of files that match the requirement of that filter

In "pure" bids-matlab it would look like:

```
BIDS = bids.layout(path_to_my_dataset)
bids.query(BIDS, 'data', opt.query)
```

So if you wanted to run your analysis on say run `02` and `05` of session `02`, you would define your filter like this:

```
opt.query.ses = '02'
opt.query.run = {'02', '05'}
```

12.7 General: What happens if we run same code twice?

In the vast majority of cases, if you have not touched anything to your options, you will overwrite the output.

Two exceptions that actually have time stamps and are not over-written:

- The `matlabbatches` saved in the `jobs` folders as `.m` and `.json` files.
- If you have saved your options with `saveOptions` (which is the case for most of bidspm actions), then the output `.json` file is saved with a time stamp too.

In most of other cases if you don't want to overwrite previous output, you will have to change the output directory.

For the preprocess action, in general you would have to specify a different `output_dir`.

For the statistics workflows, you have a few more choices as the name of the output folders includes information that comes from the options and / or the BIDS stats model.

The output folder name (generated by `getFFXdir()` for the subject level and by `getRFXdir()` for the dataset level) should include:

- the FWHM used on the BOLD images
- info specified in the `Inputs` section of the BIDS stats model JSON file (like the name of the task or the MNI space of the input images)
- the name of the run level node if it is not one of the default one (like `"run level"`).

```
$ ls demos/MoAE/outputs/derivatives/bidspm-stats/sub-01/stats

# Folder name for a model on the auditory task in SPM's MNI space
# on data smoothed with a 6mm kernel
task-auditory_space-IXI549Space_FWHM-6
```

See *this question for more details*.

12.8 Results: How can I change which slices are shown in a montage?

> In `bidsspm(..., 'action', 'results', ...)` I get an image with the overlay of different slices. How can I change which slices are shown?

When you define your options the range of slices that are to be shown can be changed like this (see `bidsResults` help section for more information):

```
% slices position in mm [a scalar or a vector]
opt.results(1).montage.slices = -12:4:60;

% slices orientation: can be 'axial' 'sagittal' or 'coronal'
% axial is default
opt.results(1).montage.orientation = 'axial';
```

12.9 SPM: How can get the data from some specific voxel in a nifti image?

12.9.1 spm_read_vols

This can be done with the `spm_read_vols` function.

```
% get header of Nifti images
header = spm_vol(path_to_image);
% get the data
data = spm_read_vols(header);
```

`data` is an array of size (x, y, z, t) with x, y and z being the spatial dimensions and t being the temporal dimension.

You can access the data in any voxel by using the proper index.

12.9.2 spm_get_data

```
% get header of Nifti images
header = spm_vol(path_to_image);
% define voxels to return
XYZ = [10 20; ...
       20 25; ...
       39 10];
% get the data
data = spm_get_data(header, XYZ)
```

- `header` - [1 x n] structure array of file headers with `n` being the number of volumes
- `XYZ` - [3 x m] or [4 x m] location matrix {voxel}
- `data` - [n x m] double values

12.10 SPM: How can I use SPM image calculator (imcalc) to create a ROI?

The image calculator utility is accessible in the SPM batch GUI:

`Batch --> SPM --> Util --> Image Calculator`

You can select the input image you want to use to create a ROI (like an atlas for example), and then set the expression you want to use to only keep certain voxels as part of the binary mask that will define your ROI.

If you want to keep voxels for your ROI that have a value of 51 or 52 or 53, you would use the following expression:

```
i1==51 || i1==52 || i1 == 53
```

If you save the batch and its job you will get an .m file that may contain something like this.

```
matlabbatch{1}.spm.util.imcalc.input = fullpath_to_the_image;
matlabbatch{1}.spm.util.imcalc.output = 'output';
matlabbatch{1}.spm.util.imcalc.outdir = {''};
matlabbatch{1}.spm.util.imcalc.expression = 'i1==51 || i1==52 || i1 == 53';
matlabbatch{1}.spm.util.imcalc.var = struct('name', {}, 'value', {});
matlabbatch{1}.spm.util.imcalc.options.dmtx = 0;
matlabbatch{1}.spm.util.imcalc.options.mask = 0;
matlabbatch{1}.spm.util.imcalc.options.interp = 1;
matlabbatch{1}.spm.util.imcalc.options.dtype = 4;
```

12.11 SPM: How can set the value of a specific voxel in a nifti image?

This can be done with the `spm_write_vols` function.

```
% get nifti header
header = spm_vol(path_to_image);

% get atlas content
content = spm_read_vols(header);

% set the voxel (1,1,1) to 0
x = 1;
y = 1;
z = 1;
content(x,y, z) = 0;

% prepare header for the output
new_header = header;
new_header.fname = 'changed_nifti.nii';

% write mask
spm_write_vol(new_header, content);
```

12.12 SPM: How do I get the header of a nifti image?

12.12.1 Getting the header

This can be done with the `spm_vol` function.

```
% get header of Nifti images
header = spm_vol(path_to_image);
```

If the input image is a 4D nifti file, the structure `header` returned will have as many elements as they are volumes in the file.

The fields of the structures are:

- `header.fname` - the filename of the image.
- `header.dim` - the x, y and z dimensions of the volume
- `header.dt` - A 1x2 array. First element is datatype (see spm_type). The second is 1 or 0 depending on the endian-ness.
- `header.pinfo` - plane info for each plane of the volume.
  - `header.pinfo(1,:)` - scale for each plane
  - `header.pinfo((2,:)` - offset for each plane The true voxel intensities of the jth image are given by: `val*header.pinfo(1,j) + header.pinfo(2,j)`
  - `header.pinfo((3,:)` - offset into image (in bytes). If the size of pinfo is 3x1, then the volume is assumed to be contiguous and each plane has the same scalefactor and offset.
- `header.mat` - a 4x4 affine transformation matrix mapping from voxel coordinates to real world coordinates. This can also be directly accessed by `spm_get_space`.

12.12.2 Voxel size

The voxel size are the 3 first values of the main diagonal of `header.mat`. Get the value along the main diagonal of the matrix and stores it in a temporary variable.

```
temp = diag(header.mat);
```

Then we only keep the 3 first values of this diagonal to get the x, y, z dimension of each voxel (in mm) of this image.

```
% the "abs" is there to have the absolute value
VoxelsDimension = abs(temp(1:3));
```

12.12.3 Getting the voxel subscripts of a given set of world space coordinates

```
% world coordinate in mm
x = 0;
y = 0;
z = 0;
world_space_coordinates = [x; y; z; 1];

% Remember that if a matrix A performs a certain transformation,
% the matrix inv(A) will perform the reverse transformation
voxel_subscripts = inv(header.mat) * world_space_coordinates;

% we need to use 'round' to get a value that is not in between 2 voxels.
voxel_subscripts = round(voxel_subscripts(1:3));
```

12.12.4 Getting the world space coordinate of a given voxel

```
x = 1;
y = 1;
z = 1;

% We have to pad with an extra one to be able
% to multiply it with the transformation matrix.
voxel_subscripts = [x; y; z; 1];
world_space_coordinates = header.mat * voxel_subscripts;

% Only the three first value of this vector are of interest to us
world_space_coordinates = world_space_coordinates(1:3);
```

12.13 SPM: How do I know which matlab function performs a given SPM process?

If you are looking for which SPM function does task X, click on the `help` button in the main SPM menu window, then on the task X (e.g Results): the new window will tell you the name of the function that performs the task you are interested in (`spm_results_ui.m` in this case).

Another tip is that usually when you run a given process in SPM, the command line will display the main function called.

For example clicking on the `Check Reg` button and selecting an image to view display:

```
SPM12: spm_check_registration (v6245)                13:42:08 - 30/10/2018
========================================================================
Display D:\SPM\spm12\canonical\avg305T1.nii,1
```

This tells you that this called the `spm_check_registration.m` matlab function.

You can also find other interesting suggestions in this discussion of the SPM mailing list: SPM: Peeking under the hood – how?.

Once you have identified the you can then type either type `help function_name` if you want some more information about this function or `edit function_name` if you want to see the code and figure out how the function works.

12.14 SPM: How do I merge 2 masks with SPM?

Here is a code snippet to merge 2 masks:

```
path_to_mask_1 = 'FIX_ME';
path_to_mask_2 = 'FIX_ME';

% get header of Nifti images
header_1 = spm_vol(path_to_mask_1);
header_2 = spm_vol(path_to_mask_2);
```

(continues on next page)

(continued from previous page)

```
% if you want to make sure that images are in the same space
% and have same resolution
masks = char({path_to_mask_1; path_to_mask_2});
spm_check_orientations(spm_vol(masks));

% get data of Nifti images
mask_1 = spm_read_vols(header_1);
mask_2 = spm_read_vols(header_2);

% concatenate data along the 4th dimension
merged_mask = cat(4, mask_1, mask_2);

% keep any voxel that has some value along the 4th dimension
merged_mask = any(merged_mask, 4);

% create a new header of the final mask
merged_mask_header = header_1;
merged_mask_header.fname = 'new_mask.nii';

spm_write_vol(merged_mask_header, merged_mask);
```

Also good way to learn about some basic low level functions of SPM.

- `spm_vol`: reads the header of a 3D or 4D Nifti images
- `spm_read_vols`: given a header it will get the data of Nifti image
- `spm_write_vol`: given a header and a 3D matrix, it will write a Nifti image

For more info about basic files, check the SPM wikibooks.

12.15 SPM: What is the content of the SPM.mat file?

This is here because SPM has the sad (and bad) Matlabic tradition of using variable names that have often attempted to replicate the notation in the papers to make engineers and the generally math enclined happy, rather than the `TypicalLongVariableNames` that many programmers and new comers would prefer to see to help with code readability.

Adapted from: http://andysbrainblog.blogspot.com/2013/10/whats-in-spmmat-file.html

12.15.1 details on experiment

- `SPM.xY.RT` - TR length (RT =”repeat time”)
- `SPM.xY.P` - matrix of file names
- `SPM.xY.VY` - (number of runs x 1) struct array of mapped image volumes (.nii file info)
- `SPM.modality` - the data you’re using (PET, FMRI, EEG)
- `SPM.stats.[modality].UFp` - critical F-threshold for selecting voxels over which the non-sphericity is estimated (if required) [default: `0.001`]

- `SPM.stats.maxres` - maximum number of residual images for smoothness estimation
- `SPM.stats.maxmem` - maximum amount of data processed at a time (in bytes)
- `SPM.SPMid` - version of SPM used
- `SPM.swd` - directory for SPM.mat and nii files. default is `pwd`

12.15.2 basis function

- `SPM.xBF.name` - name of basis function
- `SPM.xBF.length` - length in seconds of basis
- `SPM.xBF.order` - order of basis set
- `SPM.xBF.T` - number of subdivisions of TR
- `SPM.xBF.T0` - first time bin (see slice timing)
- `SPM.xBF.UNITS` - options: `'scans'` or `'secs'` for onsets
- `SPM.xBF.Volterra` - order of convolution
- `SPM.xBF.dt` - length of time bin in seconds
- `SPM.xBF.bf` - basis set matrix

12.15.3 Session structure

Note that in SPM lingo sessions are equivalent to a runs in BIDS.

user-specified covariates/regressors

e.g. motion

- `SPM.Sess([session]).C.C` - (n x c) double regressor (`c` is number of covariates, `n` is number of sessions)
- `SPM.Sess([session]).C.name` - names of covariates

conditions & modulators specified

i.e. input structure array

- `SPM.Sess([session]).U(condition).dt`: - time bin length (seconds)
- `SPM.Sess([session]).U(condition).name` - names of conditions
- `SPM.Sess([session]).U(condition).ons` - onset for condition's trials
- `SPM.Sess([session]).U(condition).dur` - duration for condition's trials
- `SPM.Sess([session]).U(condition).u` - (t x j) inputs or stimulus function matrix
- `SPM.Sess([session]).U(condition).pst` - (1 x k) peri-stimulus times (seconds)

parameters/modulators specified

- `SPM.Sess([session]).U(condition).P` - parameter structure/matrix
- `SPM.Sess([session]).U(condition).P.name` - names of modulators/parameters
- `SPM.Sess([session]).U(condition).P.h` - polynomial order of modulating parameter (order of polynomial expansion where 0 is none)
- `SPM.Sess([session]).U(condition).P.P` - vector of modulating values
- `SPM.Sess([session]).U(condition).P.P.i` - sub-indices of `U(i).u` for plotting

scan indices for sessions

- `SPM.Sess([session]).row`

effect indices for sessions

- `SPM.Sess([session]).col`

F Contrast information for input-specific effects

- `SPM.Sess([session]).Fc`
- `SPM.Sess([session]).Fc.i` - F Contrast columns for input-specific effects
- `SPM.Sess([session]).Fc.name` - F Contrast names for input-specific effects
- `SPM.nscan([session])` - number of scans per session (or if e.g. a t-test, total number of con*.nii files)

12.15.4 global variate/normalization details

- `SPM.xGX.iGXcalc` - either `'none'` or `'scaling'`

For fMRI usually is `none` (no global normalization). If global normalization is `scaling`, see `spm_fmri_spm_ui` for parameters that will then appear under `SPM.xGX`.

12.15.5 design matrix information

- `SPM.xX.X` - design matrix (raw, not temporally smoothed)
- `SPM.xX.name` - cellstr of parameter names corresponding to columns of design matrix
- `SPM.xX.I` - (nScan x 4) matrix of factor level indicators. first column is the replication number. Other columns are the levels of each experimental factor.
- `SPM.xX.iH` - vector of H partition (indicator variables) indices
- `SPM.xX.iC` - vector of C partition (covariates) indices
- `SPM.xX.iB` - vector of B partition (block effects) indices
- `SPM.xX.iG` - vector of G partition (nuisance variables) indices
- `SPM.xX.K` - cell. low frequency confound: high-pass cutoff (seconds)
- `SPM.xX.K.HParam` - low frequency cutoff value

- SPM.xX.K.X0 - cosines (high-pass filter)
- SPM.xX.W - Optional whitening/weighting matrix used to give weighted least squares estimates (WLS). If not specified spm_spm will set this to whiten the data and render the OLS estimates maximum likelihood i.e. W*W' inv(xVi.V).
- SPM.xX.xKXs - space structure for KWX, the 'filtered and whitened' design matrix
  - SPM.xX.xKXs.X - matrix of trials and betas (columns) in each trial
  - SPM.xX.xKXs.tol - tolerance
  - SPM.xX.xKXs.ds - vectors of singular values
  - SPM.xX.xKXs.u - u as in X u*diag(ds)*v'
  - SPM.xX.xKXs.v - v as in X u*diag(ds)*v'
  - SPM.xX.xKXs.rk - rank
  - SPM.xX.xKXs.oP - orthogonal projector on X
  - SPM.xX.xKXs.oPp - orthogonal projector on X'
  - SPM.xX.xKXs.ups - space in which this one is embedded
  - SPM.xX.xKXs.sus - subspace
- SPM.xX.pKX - pseudoinverse of KWX, computed by spm_sp
- SPM.xX.Bcov - xX.pKX$xX.V$xX.pKX - variance-covariance matrix of parameter estimates (when multiplied by the voxel-specific hyperparameter ResMS of the parameter estimates (ResSS/xX.trRV ResMS))
- SPM.xX.trRV - trace of R*V
- SPM.xX.trRVRV - trace of RVRV
- SPM.xX.erdf - effective residual degrees of freedom (trRV^2/trRVRV)
- SPM.xX.nKX - design matrix (xX.xKXs.X) scaled for display (see spm_DesMtx('sca',... for details)
- SPM.xX.sF - cellstr of factor names (columns in SPM.xX.I, i think)
- SPM.xX.D - struct, design definition
- SPM.xX.xVi - correlation constraints (see non-sphericity below)
- SPM.xC - struct. array of covariate info

12.15.6 header info

- SPM.P - a matrix of filenames
- SPM.V - a vector of structures containing image volume information.
  - SPM.V.fname - the filename of the image.
  - SPM.V.dim - the x, y and z dimensions of the volume
  - SPM.V.dt - a (1 x 2) array. First element is datatype (see spm_type). The second is 1 or 0 depending on the endian-ness.
  - SPM.V.mat - a (4 x 4) affine transformation matrix mapping from voxel coordinates to real world coordinates.
  - SPM.V.pinfo - plane info for each plane of the volume.
  - SPM.V.pinfo(1,:) - scale for each plane

  - SPM.V.pinfo(2,:) - offset for each plane The true voxel intensities of the jth image are given by: val*V.pinfo(1,j) + V.pinfo(2,j)
  - SPM.V.pinfo(3,:) - offset into image (in bytes). If the size of pinfo is 3x1, then the volume is assumed to be contiguous and each plane has the same scale factor and offset.

12.15.7 structure describing intrinsic temporal non-sphericity

- SPM.xVi.I - typically the same as SPM.xX.I
- SPM.xVi.h - hyperparameters
- SPM.xVi.V - xVi.h(1)*xVi.Vi{1} + ...
- SPM.xVi.Cy - spatially whitened (used by ReML to estimate h)
- SPM.xVi.CY - <(Y -)*(Y -)'> (used by spm_spm_Bayes)
- SPM.xVi.Vi - array of non-sphericity components
  - defaults to {speye(size(xX.X,1))} - i.i.d.
  - specifying a cell array of constraints ((Qi)
  - These constraints invoke spm_reml to estimate hyperparameters assuming V is constant over voxels that provide a high precise estimate of xX.V
- SPM.xVi.form - form of non-sphericity (either 'none' or 'AR(1)' or 'FAST')
- SPM.xX.V - Optional non-sphericity matrix. CCov(e)sigma^2*V. If not specified spm_spm will compute this using a 1st pass to identify significant voxels over which to estimate V. A 2nd pass is then used to re-estimate the parameters with WLS and save the ML estimates (unless xX.W is already specified).

12.15.8 filtering information

- SPM.K - filter matrix or filtered structure
  - SPM.K(s) - structure array containing partition-specific specifications
  - SPM.K(s).RT - observation interval in seconds
  - SPM.K(s).row - row of Y constituting block/partitions
  - SPM.K(s).HParam - cut-off period in seconds
  - SPM.K(s).X0 - low frequencies to be removed (DCT)
- SPM.Y - filtered data matrix

12.15.9 masking information

- SPM.xM - Structure containing masking information, or a simple column vector of thresholds corresponding to the images in VY.
- SPM.xM.T - (n x 1) double - Masking index
- SPM.xM.TH - (nVar x nScan) matrix of analysis thresholds, one per image
- SPM.xM.I - Implicit masking (0 –> none; 1 –> implicit zero/NaN mask)
- SPM.xM.VM - struct array of mapped explicit mask image volumes

- SPM.xM.xs - (1 x 1) struct ; cellstr description

12.15.10 design information

self-explanatory names, for once

- SPM.xsDes.Basis_functions - type of basis function
- SPM.xsDes.Number_of_sessions
- SPM.xsDes.Trials_per_session
- SPM.xsDes.Interscan_interval
- SPM.xsDes.High_pass_Filter
- SPM.xsDes.Global_calculation
- SPM.xsDes.Grand_mean_scaling
- SPM.xsDes.Global_normalisation

12.15.11 details on scanner data

e.g. smoothness

- SPM.xVol - structure containing details of volume analyzed
  - SPM.xVol.M - (4 x 4) voxel –> mm transformation matrix
  - SPM.xVol.iM - (4 x 4) mm –> voxel transformation matrix
  - SPM.xVol.DIM - image dimensions - column vector (in voxels)
  - SPM.xVol.XYZ - (3 x S) vector of in-mask voxel coordinates
  - SPM.xVol.S - Lebesgue measure or volume (in voxels)
  - SPM.xVol.R - vector of resel counts (in resels)
  - SPM.xVol.FWHM - Smoothness of components - FWHM, (in voxels)

12.15.12 info on beta files

- SPM.Vbeta - struct array of beta image handles
  - SPM.Vbeta.fname - beta nii file names
  - SPM.Vbeta.descrip - names for each beta file

12.15.13 info on variance of the error

- `SPM.VResMS` - file struct of ResMS image handle
  - `SPM.VResMS.fname` - variance of error file name

12.15.14 info on mask

- `SPM.VM` - file struct of Mask image handle
  - `SPM.VM.fname` - name of mask nii file

12.15.15 contrast details

added after running contrasts

- `SPM.xCon` - Contrast definitions structure array. See also `spm_FcUtil.m` for structure, rules & handling.
  - `SPM.xCon.name` - Contrast name
  - `SPM.xCon.STAT` - Statistic indicator character (`'T'`, `'F'` or `'P'`)
  - `SPM.xCon.c` - Contrast weights (column vector contrasts)
  - `SPM.xCon.X0` - Reduced design matrix data (spans design space under Ho)
    - Stored as coordinates in the orthogonal basis of xX.X from spm_sp (Matrix in SPM99b)
    - Extract using X0 `spm_FcUtil('X0', ...`
  - `SPM.xCon.iX0` - Indicates how contrast was specified:
    - If by columns for reduced design matrix then iX0 contains the column indices.
    - Otherwise, it's a string containing the `spm_FcUtil` 'Set' action: Usually one of `{'c','c+','X0'}` defines the indices of the columns that will not be tested. Can be empty.
  - `SPM.xCon.X1o` - Remaining design space data (X1o is orthogonal to X0)
    - Stored as coordinates in the orthogonal basis of xX.X from `spm_sp` (Matrix in SPM99b)
    - Extract using X1o `spm_FcUtil('X1o', ...`
  - `SPM.xCon.eidf` - Effective interest degrees of freedom (numerator df)
    - Or effect-size threshold for Posterior probability
  - `SPM.xCon.Vcon` - Name of contrast (for 'T's) or ESS (for 'F's) image
  - `SPM.xCon.Vspm` - Name of SPM image

12.16 Statistics: How can change the name of the folder of the subject level analysis?

This can be done by changing the `Name` of the run level `Nodes` in the BIDS stats model.

If your `Nodes.Name` is one of the "default" values:

- `"run"`
- `"run level"`
- `"run_level"`
- `"run-level"`
- ...

like in the example below

```
  "Nodes": [
    {
      "Level": "Run",
      "Name": "run_level",
    ...
```

then `src.stats.subject_level.getFFXdir.m()` will set the subject level folder to be named as follow:

```
sub-subLabel
└── task-taskLabel_space-spaceLabel_FWHM-FWHMValue
```

However if your `Nodes.Name` is not one of the "default" values, like this

```
  "Nodes": [
    {
      "Level": "Run",
      "Name": "parametric",
    ...
```

then the subject level folder to be named as follow:

```
sub-subLabel
└── task-taskLabel_space-spaceLabel_FWHM-FWHMValue_node-parametric
```

12.17 Statistics: How can I see what the transformation in the BIDS stats model will do to my events.tsv?

You can use the `bids.util.plot_events` function to help you visualize what events will be used in your GLM.

If you want to vivualize the events file:

```
bids.util.plot_events(path_to_events_files);
```

This assumes the events are listed in the `trial_type` column (though this can be changed by the `trial_type_col` parameter).

If you want to see what events will be included in your GLM after the transformations are applied:

```
bids.util.plot_events(path_to_events_files, 'model_file', path_to_bids_stats_model_file);
```

This assumes the transformations to apply are in the root node eof the model.

In case you want to save the output after the transformation:

```
% load the events and the stats model
data = bids.util.tsvread(path_to_events_files);
model = BidsModel('file', path_to_bids_stats_model_file);

% apply the transformation
transformers = model.Nodes{1}.Transformations.Instructions;
[new_content, json] = bids.transformers(transformers, data);

% save the new TSV for inspection to make sure it looks like what we expect
bids.util.tsvwrite(fullfile(pwd, 'new_events.tsv'), new_content);
```

12.18 Statistics: How should I name my conditions in my events.tsv?

In BIDS format, conditions should be named in the `trial_type` column of the `events.tsv` file.

Some good practices for naming "things" can probably be applied here.

- use only alphanumeric characters, underscores and hyphens, avoid spaces in the condition names

For example: `foo_bar` is ok, but `f$o}o b^a*r` is not.

- the condition names should be short AND descriptive AND human readable

For example: `1` or `one` or `condition1` are short but not descriptive.

An extra requirement to have condition names that can work with bidspm is that condition names ending with an underscore followed by one or more digits may lead to unwanted behavior (error or nothing happening

- see issue) when computing results of a GLM.

So for example:

`happy_face1` and `house123` are ok, but `happy_face_1` and `house_123` are not.

If your BIDS dataset has conditions that do not follow this rule, then you can use the `Replace` variable transform in your BIDS statistical model to rename them on the fly without having to manually edit potentially dozens of files.

See also example below.

Say one of your events.tsv files looks like this:

onset	duration	trial_type
2	2	happy_face_1
4	2	house_2
5	2	happy_face_2
8	2	house_4

You can add the following transformation to the first node (usually the run level node) of your BIDS stats model.

```
{
  "Nodes": [
    {
      "Level": "Run",
      "Name": "run_level",
      "GroupBy": [
        "run",
        "subject"
      ],
      "Transformations": {
        "Description": "Replace",
        "Instruction": [
            {
                "Name": "Replace",
                "Input": "trial_type",
                "Replace": [
                    {
                        "key": "happy_face_1",
                        "value": "happy_face1"
                    },
                    {
                        "key": "house_2",
                        "value": "house2"
                    },
                    {
                        "key": "happy_face_2",
                        "value": "happy_face2"
                    },
                    {
                        "key": "house_4",
                        "value": "house4"
                    }
                ]
            }
        ]
    },
      "Model": {
        "X": [
            "trial_type.happy_face*",
            "trial_type.house*"
          ],
        "Description": "the rest of the bids stats model would go below this as usual."

    }
    }
  ]
}
```

Related issue: https://github.com/cpp-lln-lab/bidspm/issues/973

12.19 Statistics: How should I structure my data to run my statistical analysis?

The main thing to remember is that bidspm will read the events.tsv files from your raw BIDS data set and will read the bold images from a `bidspm-preproc` folder.

If your data was preprocessed with fmriprep, bidspm will first need to copy, unzip and smooth the data into a `bidspm-preproc` folder

Here is an example of how the data is organized for the MoAE fmriprep demo and what the `bidspm` BIDS call would look like.

```
├── inputs
│   ├── fmriprep                        # fmriprep preprocessed BIDS dataset
│   │   ├── dataset_description.json
│   │   └── sub-01
│   │       ├── anat
│   │       ├── figures
│   │       └── func
│   │           ├── sub-01_task-auditory_desc-confounds_timeseries.json
│   │           ├── sub-01_task-auditory_desc-confounds_timeseries.tsv
│   │           ├── sub-01_task-auditory_space-MNI152NLin6Asym_desc-brain_mask.json
│   │           ├── sub-01_task-auditory_space-MNI152NLin6Asym_desc-brain_mask.nii.gz
│   │           ├── sub-01_task-auditory_space-MNI152NLin6Asym_desc-preproc_bold.json
│   │           └── sub-01_task-auditory_space-MNI152NLin6Asym_desc-preproc_bold.nii.gz
│   └── raw                             # raw BIDS dataset
│       ├── dataset_description.json
│       ├── README
│       ├── sub-01
│       │   ├── anat
│       │   │   └── sub-01_T1w.nii
│       │   └── func
│       │       ├── sub-01_task-auditory_bold.nii
│       │       └── sub-01_task-auditory_events.tsv
│       └── task-auditory_bold.json
├── models                              # models used to run the GLM
│   ├── model-MoAEfmriprep_smdl.json
│   ├── model-MoAEindividual_smdl.json
│   └── model-MoAE_smdl.json
├── options
└── outputs
    └── derivatives
        └── bidspm-preproc              # contains data taken from fmriprep and smoothed
            ├── dataset_description.json
            ├── README
            ├── jobs
            │   └── auditory
            │       └── sub-01
            └── sub-01
                ├── anat
                └── func
                    ├── sub-01_task-auditory_desc-confounds_timeseries.json
                    ├── sub-01_task-auditory_desc-confounds_timeseries.tsv
```

(continues on next page)

(continued from previous page)

```
                    ├── sub-01_task-auditory_space-MNI152NLin6Asym_desc-brain_mask.json
                    ├── sub-01_task-auditory_space-MNI152NLin6Asym_desc-brain_mask.nii
                    ├── sub-01_task-auditory_space-MNI152NLin6Asym_desc-preproc_bold.json
                    ├── sub-01_task-auditory_space-MNI152NLin6Asym_desc-preproc_bold.nii
                    ├── sub-01_task-auditory_space-MNI152NLin6Asym_desc-smth8_bold.json
                    └── sub-01_task-auditory_space-MNI152NLin6Asym_desc-smth8_bold.nii
```

```
WD = fileparts(mfilename('fullpath'));

subject_label = '01';

bids_dir = fullfile(WD, 'inputs', 'raw');
output_dir = fullfile(WD, 'outputs', 'derivatives');
preproc_dir = fullfile(output_dir, 'bidspm-preproc');

model_file = fullfile(pwd, 'models', 'model-MoAEfmriprep_smdl.json');

bidspm(bids_dir, output_dir, 'subject', ...
        'participant_label', {subject_label}, ...
        'action', 'stats', ...
        'preproc_dir', preproc_dir, ...
        'model_file', model_file, ...
        'fwhm', 8, ...
        'options', opt);
```

Generated by FAQtory

CHAPTER

THIRTEEN

FIELDMAPS

In a nutshell, the information we need to create a VDM in SPM (see `calculate VDM` module in SPM batch):

- `blip direction`
- `echo time`
- `total EPI readout time`

Inferring `blip direction` and `echo time` from a dataset that has sufficient metadata is usually simple.

But `total EPI readout time` is not mentioned, so it has to be computed from the information we have, it is not entirely clear how (see the comments with a lot of `???` in `getTotalReadoutTime`).

Things that are yet unclear:

- is it actually possible to compute total EPI readout time that SPM needs from the info in a typical dataset with fieldmaps like `openneuro/ds001168`?
- If it is not then that is an issue because it means some BIDS dataset are not usable with SPM.

Things to keep an eye on: the code from this repo from the fMRIprep team could have answers for us.

bidsCreateVDM()

setBatchCoregistrationFmap() *setBatchCreateVDMs()* *setBatchComputeVDM()*

getBlipDirection() *getMetadataFromIntendedForFunc()* *getTotalReadoutTime()* *getVdmFile()*

CHAPTER

FOURTEEN

QUALITY CONTROL

Note: The illustrations in this section mix what the files created by each workflow and the functions and are called by it. In this sense they are not pure DAGs (directed acyclic graphs) as the `*.m` files mentioned in them already exist.

- `anatomicalQA()`
- `functionalQA()`

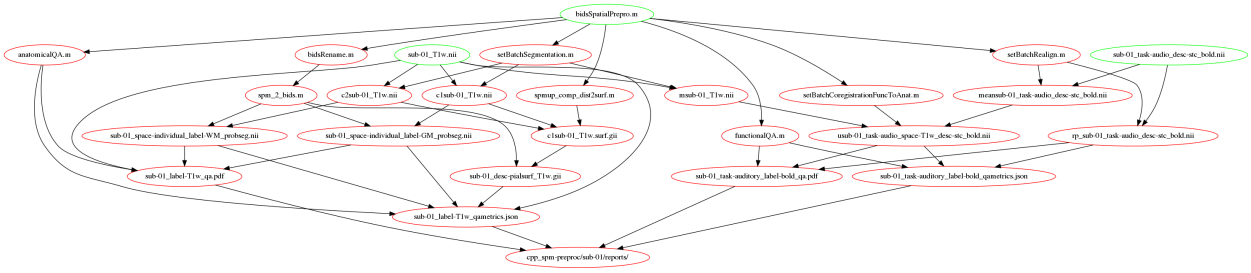


Fig. 1: workflows for QA as part of the spatial preprocessing workflow

- *computeDesignEfficiency()*
- *plotEvents()*

CHAPTER

FIFTEEN

MANUAL COREGISTRATION

Manual coregistration tools

---

mancoreg(*varargin*)

This function displays 2 SPM ortho-views of a `targetimage` and a `sourceimage` image that can be manually coregistered.

USAGE:

```
mancoreg('targetimage', [], 'sourceimage', [], 'stepsize', 0.01)
```

Parameters

- **targetimage** (`string`) – Filename or fullpath of the target image. If none is provided you will be asked by SPM to select one.
- **sourceimage** (`string`) – Filename or fullpath of the source image. If none is provided you will be asked by SPM to select one.
- **stepsize** (`positive float`) – step size for each rotation and translation

Manual coregistration tool

The source image (bottom graph) can be manually rotated and translated with 6 slider controls. In the source graph the source image can be exchanged with the target image using a radio button toggle. This is helpful for visual fine control of the coregistration. The transformation matrix can be applied to a selected set of volumes with the `apply transformation` button. If the transformation is to be applied to the original source file that file will also need to be selected. If the sourceimage or targetimage are not passed the user will be prompted with a file browser.

The code is loosely based on `spm_image()` and `spm_orthoviews()` It requires the m-file with the callback functions for the user controls (`mancoregCallbacks()`).

(C) Copyright 2004-2009 JH (C) Copyright 2009_2012 DSS (C) Copyright 2012_2019 Remi Gau (C) Copyright 2020 CPP BIDS SPM-pipeline developers

mancoregCallbacks(*operation*)

Callback routines for `mancoreg()`: defines the different actions for the different buttons.

USAGE:

```
mancoreg_callbacks(operation)
```

Parameters

operation (string) – Can be any of the following: move, toggle_off, toggle_on, reset, apply, plotmat

CHAPTER

SIXTEEN

DOCKER

16.1 Get image from docker hub

```
docker pull cpplab/bidspm:latest
```

16.2 Build the docker image locally

If you want to build the docker image locally and not pull it from the docker hub:

```
docker build . -f docker/Dockerfile -t cpplab/bidspm:latest
```

This will create an image with the tag name `bidspm:latest`

Running `make build_image` will also build the `stable` version and a version tagged image.

16.3 Run the docker image

The following command would pull from our docker hub and start the docker image:

```
.. code-block:: bash
```

docker run -it –rm cpplab/bidspm:latest

The following command would do do the same, but it would also map 2 folders from your computer and run preprocessing on your dataset (assuming there is a task called `auditory`):

```
bids_dir=fullpath_to_bids_dataset
output_dir=fullpath_to_your_output_folder

docker run -it --rm \
    -v $bids_dir:/raw \
    -v $output_dir:/derivatives \
    cpplab/bidspm:latest \
        /raw \
        /derivatives \
        subject \
        --task auditory \
```

(continues on next page)

(continued from previous page)

```
        --action preprocess \
        --fwhm 8
```

Similarly this would run the statistics provided you give a bids stats model file and the path to the preprocessed data.

```
bids_dir=fullpath_to_bids_dataset
output_dir=fullpath_to_your_output_folder
model=fullpath_to_your_bids_stats_model

docker run -it --rm \
    -v $bids_dir:/raw \
    -v $output_dir:/derivatives \
    -v $model:/models/smdl.json \
    cpplab/bidspm:latest \
        /raw \
        /derivatives \
        subject \
        --action stats \
        --preproc_dir /derivatives/bidspm-preproc \
        --model_file /models/smdl.json \
        --fwhm 8
```

CHAPTER

SEVENTEEN

CHANGELOG

All notable changes to this project will be documented in this file.

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

17.1 [Unreleased]

17.1.1 Added

- [ENH] allow to copy anat only on raw datasets #1181 by @Remi-Gau
- [ENH] add option to concatenate runs at subject level to facilite running PPI analysis #1133 by @Remi-Gau
- [ENH] allow to run substeps of substeps of the bayesian model selection #1145 by @Remi-Gau
- [ENH] add quality control for GLM using the MACS toolbox to give a goodness of fit and several other information criteria (AIC, BIC) (MATLAB only) #1135 by @Remi-Gau
- [ENH] add several confound strategies to add to bids stats model and a function to create family of models #1126 by @Remi-Gau
- [ENH] add CLI to run bayesian model selection #1121 by @Remi-Gau
- [ENH] support label of activations with all atlases 1100 by Remi-Gau
- [ENH] add support for session level models #1116 be @Remi-Gau
  - allow to pass dummy contrasts to session level
  - allow inter session contrasts to be computed at the subject level
  - improve naming of contrast to mention bids ses and run

17.1.2 Changed

- [REF] Refactor and update CLI in #1096 @Remi-Gau

17.1.3 Deprecated

17.1.4 Removed

- [REF] Remove old CLI in #1096 @Remi-Gau

17.1.5 Fixed

- [FIX] copy the MACS toolbox to the SPM toolbox folder during the initialisation #1203 by @Remi-Gau
- [FIX] save onsets.mat directly in subject stats folder #1187 by @Remi-Gau
- [FIX] do not compute subject level contrast when running dataset level #1102 by @Remi-Gau
- [FIX] copy `RepetitionTime` in sidecar JSON after running smoothing in #1099 by @Remi-Gau
- [FIX] rename results files (csv, tsv, png, nii) of each contrasts #1104 by @Remi-Gau
- [DIX] reslice ROIS before running ROI based analysis to make sure they are at the resolution of the BOLD images in #1110 by @Remi-Gau

17.1.6 Security

17.2 [3.1.0] - 2023-07-01

Full Changelog: https://github.com/cpp-lln-lab/bidspm/compare/v3.0.0...v3.1.0

17.2.1 Added

- [DOC] add changelog in 1056 by Remi-Gau
- [DOC] minor doc fixes in 1020 by Remi-Gau
- [DOC] Improve doc example out stats in 1014 by Remi-Gau
- [DOC] update doc regarding FAST model as default in 985 by Remi-Gau
- [DOC] update contributors and default options in doc in 981 by Remi-Gau
- [DOC] add doc to link to transformations in 942 by Remi-Gau
- [DOC] add demo code from workshop in 935 by Remi-Gau
- [DOC] abstract OHBM 2023 in 934 by Remi-Gau
- [DOC] add argument groups to python cli in 907 by Remi-Gau
- [DOC] update FAQ in 897 by Remi-Gau
- [ENH] add proper error when a column cannot be found in participants.tsv in 1050 by Remi-Gau
- [ENH] update CPP ROI and add methods section for ROI creation in 1026 by Remi-Gau
- [ENH] update rename to store original spm name of a file in metadata in 987 by Remi-Gau
- [ENH] add hemisphere parameter to CLI in 950 by Remi-Gau
- [ENH] make it possible to update bidspm from any folder in 948 by Remi-Gau
- [ENH] use filtering of layout in 944 by Remi-Gau

- [ENH] make reports more silent in 939 by Remi-Gau
- [ENH] add CC0 license by default to all outputs in 898 by Remi-Gau

17.2.2 Changed

- [ENH] overwrite files when renaming by default in 1051 by Remi-Gau
- [ENH] turn error into warning when no data to copy found in 992 by Remi-Gau

17.2.3 Removed

- [ENH] drop rsHRF support in 906 by Remi-Gau

17.2.4 Fixed

- [FIX] fix vismotion demo in 1070 by Remi-Gau
- [FIX] Exclude from GLM specification events with onsets longer than the run duration in 1060 by Remi-Gau
- [FIX] remove dummies from preproc dataset and not raw dataset when using CLI in 1057 by Remi-Gau
- [FIX] skip smoothing when running bidspm prepoc in dryRun in 1054 by Remi-Gau
- [FIX] handle phase entity in filename in 1034 by Remi-Gau
- [FIX] fix group level results after contrasts smoothing in 1021 by Remi-Gau
- [FIX] copy to derivatives handles bids filter file and minimize re copying files that already exist in 1015 by Remi-Gau
- [FIX] allow cli to run constrat smoothing in 1012 by Remi-Gau
- [FIX] Force copy of data from fmriprep folder even if bidspm-preproc folder exists in 1009 by marcobarilari
- [FIX] report proper fold number in labelfold.tsv in 989 by Remi-Gau
- [FIX] inconsistent slice timing throw errors and not warnings in 982 by Remi-Gau
- [FIX] validate condition names early to avoid downstream error in 983 by Remi-Gau
- [FIX] validate content results structure in 980 by Remi-Gau
- [FIX] missing variable for a Filter transform should not lead to a crash in 970 by Remi-Gau
- [FIX] display error when impossible slice timing values are given in 969 by Remi-Gau
- [FIX] fix QA failures in 941 by Remi-Gau
- [FIX] bug fix 892 in 936 by Remi-Gau
- [FIX] fix python cli in 888 by Remi-Gau

17.3 [3.0.0] - 2022-12-14

Full Changelog: https://github.com/cpp-lln-lab/bidspm/compare/v2.3.0...v3.0.0

17.3.1 Changed

- [DEP] deprecate slice order in options in 882 by Remi-Gau
- [ENH] change the way parametric models are run in 873 by Remi-Gau*

17.3.2 Added

- [ENH] include roi based calls in CLI in 880 by Remi-Gau
- [ENH] add copy dataset to CLI in 842 by Remi-Gau
- [ENH] add smoothing to CLI in 841 by Remi-Gau
- [ENH] create docker image of bids app - Octave in 837 by Remi-Gau
- [ENH] use python based CLI to run bidspm with octave in 832 by Remi-Gau
- [ENH] update CPP_ROI in 885 by Remi-Gau
- [ENH] add extra files to derivatives datasets in 883 by Remi-Gau
- [ENH] incorporate opt.results in bids stats model in 879 by Remi-Gau
- [ENH] add option to ignore creating dataset level node in default model in 871 by Remi-Gau
- [ENH] implement logger in 867 by Remi-Gau
- [ENH] add function to return contrast filename for a certain contrast name in 866 by Remi-Gau
- [DOC] update FAQ to explain how to change subject level GLM folder name in 872 by Remi-Gau

17.3.3 Fixed

- [FIX] better handle metadata when changing suffix in 884 by Remi-Gau
- [FIX] exit with a warning and not an error when no contrast specified in 870 by Remi-Gau
- [FIX] fix SPM loading in returnContrastImageFile in 869 by Remi-Gau
- [FIX] fix make file in 868 by Remi-Gau

17.4 [2.3.0] - 2022-11-22

Full Changelog: https://github.com/cpp-lln-lab/bidspm/compare/v2.2.0...v2.3.0

- `bidspm` main function:
  - saving options are saved to help with bug report
  - generate method section in reports folder when running preprocess or statistics

17.4.1 Added

- [DOC] add auto label of activation info in 821 by Remi-Gau
- [DOC] improve bids model warning in 820 by Remi-Gau
- [DOC] add all functions to doc in 819 by Remi-Gau
- [ENH] add design only to CLI 772
- [ENH] smoothing workflow will also try to smooth the corresponding anat data too
- [ENH] save skipped ROIs and concat beta maps in 823 by Remi-Gau
- [ENH] add boilerplate to CLI in 822 by Remi-Gau
- [ENH] error logs are generated upon crash for better bug reports in 808 by Remi-Gau
- [ENH] Use CLI to create default model in 804 by Remi-Gau
- [ENH] include bids and bids stats model validation if the validators are installed in 787 by Remi-Gau
- [ENH] add inverse normalize workflow in 784 by Remi-Gau
- [ENH] lesion detection will be done by including the CSF TPM too in 778 by Remi-Gau

17.4.2 Fixed

- [FIX] fix printing of windows path in 812 by Remi-Gau
- [FIX] use glob patterns to define dummy contrasts in 826 by Remi-Gau
- [FIX] rm desc when renamining some files in lesion segemtation in 817 by Remi-Gau
- [FIX] avoid duplicate contrasts in 816 by Remi-Gau
- [FIX] make collecting of OS version on windows more robust in 806 by Remi-Gau
- [FIX] throw warning when no results are asked in 795 by Remi-Gau
- [FIX] fix windows bugs in 792 by Remi-Gau
- [FIX] fix spelling in 789 by Remi-Gau
- [FIX] fix and tidy failing workflows in 774 by Remi-Gau

17.5 [2.2.0] - 2022-10-29

Full Changelog: https://github.com/cpp-lln-lab/bidspm/compare/v2.1.0...v2.2.0

17.5.1 Added

- [DOC] use new copyright format in 750 by Remi-Gau
- [DOC] improve stats doc and warnings in 746 by Remi-Gau
- [ENH] add design only to CLI in 772 by Remi-Gau
- [ENH] add functions to help select fmriprep regressors in 748 by Remi-Gau
- [ENH] start switching to bidspm in 747 by Remi-Gau

- [ENH] use inputs from several datasets for lesion abnormality detection in 730 by Remi-Gau
- [ENH] Update bidspm path in 752 by Remi-Gau

17.5.2 Changed

- [DEP] update bids matlab in 734 by Remi-Gau

17.5.3 Fixed

- [FIX] fixes workflow timing in 773 by Remi-Gau
- [FIX] related to 763 and testing through CI if the patch breaks things (reopened) in 765 by marcobarilari

17.6 [2.1.0] - 2022-07-21

Full Changelog: https://github.com/cpp-lln-lab/bidspm/compare/v2.0.0...v2.1.0

17.6.1 Added

- [ENH] add possibility to use AAL for labelling of activations in 717 by Remi-Gau
- [DOC] update FAQ to help structure data to run stats on fmriprep in 685 by Remi-Gau
- [DOC] add demo for ds002799 in 678 by Remi-Gau
- [DOC] add DIY section to documtation in 677 by Remi-Gau
- [DOC] improve description of stats output and of bids stats model in 670 by Remi-Gau

17.6.2 Changed

- [MNT] update citation file in 721 by Remi-Gau
- [DEP] update bids matlab in 680 by Remi-Gau

17.6.3 Fixed

- [FIX] fix lesion segmentation output (and refactor) in 727 by Remi-Gau
- [FIX] add more explicit error message for input parsing of getData in 726 by Remi-Gau
- [FIX] support globbing patterns to specify conditions in design matrix in 716 by Remi-Gau
- [FIX] fix ALI toolbox issues in 723 by Remi-Gau
- [FIX] improve warning selecting too many masks in 715 by Remi-Gau
- [FIX] fix and refactor demos, and update help sections in 701 by Remi-Gau
- [FIX] add guard clauses to run ROI based analysis only when requested in 708 by Remi-Gau
- [FIX] convert nan to zeros in confounds in 700 by Remi-Gau
- [FIX] change default space value in cpp_spm in 699 by Remi-Gau

- [FIX] various bug fixes in 694 by Remi-Gau
- [FIX] add some warnings to handle several bugs with empty ROIs in 693 by Remi-Gau
- [FIX] re enable octave tests CI in 686 by Remi-Gau
- [FIX] patches for 682 and 683 (#687) by mwmaclean
- [FIX] ensure that bidsResults does not run if we don't have the proper options in 679 by Remi-Gau
- [FIX] returnRootDir only relies on fullpaths in 676 by Remi-Gau
- [FIX] set minimum compatible fmriprep version in 675 by Remi-Gau
- [FIX] renaming of design matrix images at group level in 668 by Remi-Gau
- [FIX] make sure 2 sample ttest can run more than one contrasts in 665 by Remi-Gau
- [FIX] label actiation: csv results file with no significant voxels are ignored with a warning in 663 by Remi-Gau
- [FIX] onsets and confounds are saved in the proper dir when there are several tasks in 659 by Remi-Gau
- [FIX] fix silent dry run override in 657 by Remi-Gau
- [FIX] use proper subject background for montage in 656 by Remi-Gau
- [FIX] allow extra BIDS entities to be integrated in glm dir name in 654 by Remi-Gau
- [FIX] make GLM path more consistent in 652 by Remi-Gau
- [FIX] prevent some crashes if Model.Input are not passed as arrays in 650 by Remi-Gau
- [FIX] fix printing of of path in 649 by Remi-Gau
- [FIX] properly skip segment and sullstrip in 638 by Remi-Gau
- [FIX] use unix format when printing path to screen in 634 by Remi-Gau
- [FIX] getEnvInfo for windows in 631 by Remi-Gau

17.7 [2.0.0] - 2022-07-10

Full Changelog: https://github.com/cpp-lln-lab/bidspm/compare/v1.1.5...v2.0.0

17.7.1 Added

- [DOC] prepare binder and basic jupyter notebook by Remi-Gau in 155
- [DOC] general doc update by Remi-Gau in 446
- [DOC] Add figures for some workflows by Remi-Gau in 463
- [DOC] adds link and references to other SPM material by Remi-Gau in 464
- [DOC] improve results doc and associated how to by Remi-Gau in 470
- [DOC] Misc doc updates by Remi-Gau in 484
- [DOC] update doc on how to filter files and what files gets upsampled by Remi-Gau in 494
- [DOC] update templates by Remi-Gau in 455
- [DOC] add MyST for the doc and add list of default options in the doc by Remi-Gau in 557
- [DOC] update RTD with a symlink to README by Remi-Gau in 569

- [DOC] fmriprep stats demo by Remi-Gau in 594
- [DOC] Update getRegressorIdx.m by Remi-Gau in 601
- [DOC] update where references are stored by Remi-Gau in 614
- [DOC] update vismotion demo by Remi-Gau in 624
- [DOC] Update FAQ and BIDS stats model related doc by Remi-Gau in 626
- [ENH] adapt workflows to new bids-matlab by Remi-Gau in 368
- [ENH] adapt to use fmriprep input with rshrf toolbox by Remi-Gau in 370
- [ENH] add verbosity control by Remi-Gau in 381
- [ENH] ROI tSNR pipeline by Remi-Gau in 401
- [ENH] allow fMRIprep input for GLM by Remi-Gau in 367
- [ENH] update dockerfiles by Remi-Gau in 420
- [ENH] Add QA functions to plot events file and compute design efficiency by Remi-Gau in 428
- [ENH] compute tSNR for a given mask by mwmaclean in 402
- [ENH] add a function to deinitialize CPP SPM and make sure there is just one instance in the path by Remi-Gau in 435
- [ENH] Allow for multi tasks processing by Remi-Gau in 439
- [ENH] Update BIDS stats model by Remi-Gau in 441
- [ENH] Enhancements subject / group level GLM and results by Remi-Gau in 443
- [ENH] add elapsedTime function by marcobarilari in 229
- [ENH] drop support for parfor loops by Remi-Gau in 447
- [ENH] reports are saved for each subject being processed by Remi-Gau in 448
- [ENH] drop nifti tools dependency by Remi-Gau in 449
- [ENH] add possibility to limit maximum number of volumes per run in a subject level GLM by Remi-Gau in 451
- [ENH] save group stats in separate derivatives folder by Remi-Gau in 453
- [ENH] improve confounds inclusion in design matrix by Remi-Gau in 454
- [ENH] add metadata consistency checks by Remi-Gau in 457
- [ENH] use BIDS stats model to select input task, space and override options by Remi-Gau in 461
- [ENH] update roi based glm by Remi-Gau in 465
- [ENH] integrates anat and func QA as part of `bidsSpatialPrepro` by Remi-Gau in 466
- [ENH] create an anat only spatial preprocessing by Remi-Gau in 467
- [ENH] add workflow to perform model selection using the MACS toolbox by Remi-Gau in 472
- [ENH] enhancements for ROI based GLM by Remi-Gau in 477
- [ENH] save jobs as m file by Remi-Gau in 482
- [ENH] filter file volume by Remi-Gau in 492
- [ENH] start implementing BIDS stats model transformers by Remi-Gau in 493
- [ENH] Replace anat reference by bids filter by Remi-Gau in 497

- [ENH] start creating main API by Remi-Gau in 511
- [ENH] change verbosity levels by Remi-Gau in 507
- [ENH] Add change suffix workflow by Remi-Gau in 516
- [ENH] run subject level GLM with no condition by Remi-Gau in 520
- [ENH] add remove dummies workflow by Remi-Gau in 521
- [ENH] transformers 2 by Remi-Gau in 522
- [ENH] rename output func qa by Remi-Gau in 533
- [ENH] stats model and results by Remi-Gau in 541
- [ENH] add workflow for creation of T1map from mp2rage by Remi-Gau in 542
- [ENH] only save batches as .m files by Remi-Gau in 559
- [ENH] create a BIDS app API by Remi-Gau in 564
- [ENH] skullstripping fixes and options by Remi-Gau in 571
- [ENH] automatically generate method sections by Remi-Gau in 572
- [ENH] fix some issue on model selection by Remi-Gau in 574
- [ENH] simplify and extend bidsResults by Remi-Gau in 577
- [ENH] adapt group level analysis to work with BIDS stats model by Remi-Gau in 581
- [ENH] add F test by Remi-Gau in 584
- [ENH] Add parametric modulation to run / subject level GLM by Remi-Gau in 585
- [ENH] use native resolution for segmentation for lesion detection by Remi-Gau in 588
- [ENH] add “force” parameter to bidsCopyInputFolder by Remi-Gau in 589
- [ENH] add metadata to preprocessed derivatives by Remi-Gau in 580
- [ENH] misc improvements at the run level GLM by Remi-Gau in 592
- [ENH] improve reports by Remi-Gau in 595
- [ENH] add two sample t test group level batch by Remi-Gau in 597
- [ENH] allow to run “contrasts” and “results” from main API by Remi-Gau in 615
- [ENH] improve group level analysis by Remi-Gau in 620
- [ENH] make it possible to run models / contrasts using other columns than trial_type by Remi-Gau in 621
- [ENH] add neuromorphometrics label to bidsResults output when in MNI space by Remi-Gau in 622

17.7.2 Changed

- [DEP] update bids matlab by Remi-Gau in 570
- [ENH] Rename preprocessing output to bids by Remi-Gau in 395
- [ENH] use official HRF model from bids stats model by Remi-Gau in 604
- [ENH] properly rely on BIDS stats model to specify subject level contrast by Remi-Gau in 576
- [ENH] Update submodules by Remi-Gau in 575
- [ENH] Bump bids matlab by Remi-Gau in 495

- [ENH] drop stats folder by Remi-Gau in 539
- [ENH] change MNI to SPM default IXI549Space by Remi-Gau in 456

17.7.3 Fixed

- [FIX] adapt to the new bids-matlab "dev" by Remi-Gau in 366
- [FIX] apply 418 to dev by Remi-Gau in 419
- [FIX] Update binder by Remi-Gau in 413
- [FIX] apply 425 to dev by Remi-Gau in 426
- [FIX] Fix facerep demo by Remi-Gau in 438
- [FIX] make sure default BIDS models are usable by Remi-Gau in 459
- [FIX] reolve issues to get anat file from a different session from the func data by Remi-Gau in 462
- [FIX] returnRootDir did not return the root dir by Remi-Gau in 498
- [FIX] throw error when no repetition time was found by Remi-Gau in 509
- [FIX] update vismotion demo by Remi-Gau in 513
- [FIX] Misc bug squashing :bug: :skull: by Remi-Gau in 515
- [FIX] Fix typos in the sh run file of vismotion demo by marcobarilari in 524
- [FIX] fix 545 add more options for segmentation batch by marcobarilari in 547
- [FIX] creates BIDS valid filename for ROIs in individual space by Remi-Gau in 562
- [FIX] fix system test and silence ALI warning by Remi-Gau in 596
- [FIX] Spatial preprocessing normalizes output of skullstripping by Remi-Gau in 602
- [FIX] fix several issues for GLM at the subject level by Remi-Gau in 606
- [FIX] fix several issues related to getting the correct files for the GLM by Remi-Gau in 607
- [FIX] GLM: allow to filter input files based on BIDS entities for bold files by Remi-Gau in 611

CHAPTER

EIGHTEEN

LINKS AND REFERENCES

18.1 SPM starters

If you start from zero, go through the 2 first tutorials of SPM

18.2 Andrew Jahn videos and blogs

- video playlist
- marsbar
- SPM
- SPM.mat

18.3 SPM code snippets

SPM wikibook has some very useful sections.

From John Ashburner on Tom Nichols blog

From Rik Henson

Some follow along tutorials written a long time ago, and that probably should be turned into notebooks and updated.

- Basic file / image manipulation with SPM
- HRF, convolution and GLM ("by hand")
- Design efficiency

18.4 Bilbiography

CHAPTER

NINETEEN

CONTRIBUTING GUIDELINES

Feel free to open issues to report a bug and ask for improvements.

If you want to contribute, have a look at our contributing guidelines that are meant to guide you and help you get started. If something is not clear or you get stuck: it is more likely we did not do good enough a job at explaining things. So do not hesitate to open an issue, just to ask for clarification.

19.1 Installation

To install bidspm and make changes to it, it is recommend to install the python package in editable mode with all its development dependencies.

```
pip install -e .[dev]
```

19.2 Initialisation

To facilitate running tests, make sure you initialize bidspm in dev mode, from the MATLAB command line:

```
bidspm dev
```

You can also run all the tests with:

```
bidspm run_tests
```

19.3 Style guidelines

We use `camelCase` to name functions and variables for the vast majority of the code in this repository.

Scripts names in general and as well functions related to the demos use a `snake_case`.

Constant are written in `UPPERCASE`.

19.3.1 Input arguments ordering

From more general to more specific

`BIDS` > `opt` > `subject` > `session` > `run`

- `BIDS` (output from `getData` or `bids.layout`) restrict the set of possible analysis one can run to this BIDS data set
- `opt` restricts this set even further
- `subject` / `session` / `run` even more

```
% OK
varargout = getInfo(BIDS, opt, subLabel, info, varargin)

% not OK
varargout = getInfo(subLabel, BIDS, opt, info, varargin)
```

19.3.2 Output arguments ordering

Try to return them in order of importance first and in order of appearance otherwise.

19.3.3 Exceptions

If function creates or modifies a batch then `matlabbatch` is the first `argin` and first `argout`.

If a function performs an "action" to be chosen from a one of several strings (with a switch statement), this string comes as first `argin` or second if `matlabbatch` is first.

```
% OK
varargout = getInfo('filename', BIDS, opt, subID, varargin)
[matlabbatch, voxDim] = setBatchRealign(matlabbatch, [action = 'realign',] BIDS, opt,␣
↪subID)

% not OK
% 'filename' is the name of the "action" or the info to get in this case
% batch and action should go first
varargout = getInfo(BIDS, opt, subID, 'filename', varargin)
[matlabbatch, voxDim] = setBatchRealign(BIDS, opt, subID, matlabbatch, [action = 'realign
↪'])
```

19.4 Updating the FAQ

The FAQ is rendered with FAQtory.

New questions must be added to `docs/questions` and the main FAQ.md can be created with:

```
make update_faq
```

19.5 release protocol

- [ ] create a dedicated branch for the release candidate
- [ ] update version in `citation.cff`
- [ ] documentation related
  - [ ] ensure the documentation is up to date
  - [ ] make sure the doc builds correctly and fix any error
- [ ] update jupyter books
- [ ] update binder
- [ ] update docker recipes
- [ ] update changelog
- [ ] run `make release`
- [ ] open a pull request (PR) from this release candidate branch targeting the default branch
- [ ] fix any remaining failing continuous integration (test, markdown and code linting…)
- [ ] merge to default branch
- [ ] create a tagged release
- [ ] build and push docker images if necessary

CHAPTER

TWENTY

DEVELOPERS DOCUMENTATION

At the highest levels bidspm is organized in workflows:

- they all start with the prefix `bids` (for example `bidsRealignReslice`)
- they are in the folder `src.workflows`
- they run on all the subjects specified in the `options` structure.

Most workflows run by creating matlab batches that are saved as `.mat` files in a `jobs` then passed to the SPM jobman to run. To do this the workflows call "batch creating functions":

- all start with the prefix `setBatch` (for example `setBatchCoregistration`).
- are in the folder `src.batches`.

Many workflows include some post-processing steps (like file renaming) after the execution of the batch, so in many cases the output of running just the batch and running the whole workflow will be different.

Preprocessing, *Statistics* and *Fieldmaps* handling have their own document pages.

Other workflows, batches and related helper functions are listed below.

20.1 workflows

bidsChangeSuffix(*varargin*)

USAGE:

```
bidsChangeSuffix(opt, newSuffix, 'filter', struct([]), 'force', false)
```

Parameters

- **opt** (`structure`) – Options chosen for the analysis. To test the output, set `opt.dryRun` to `true`. See checkOptions..
- **newSuffix** (`char`) – TODO: add checks on newSuffix to make sure it only contains [a-zA-Z0-9]
- **filter** (`structure`) – structure to decide which files to include. Default: `struct([])` for no filter
- **force** (`boolean`) – If set to `true` it will overwrite already existing files. Default: `false`

EXAMPLE:

```
opt.dir.input = path_to_dataset;
opt.dryRun = true;

newSuffix = 'vaso';

filter = struct('suffix', 'bold');

bidsChangeSuffix(opt, newSuffix, ...
                 'filter', filter, ...
                 'force', false)
```

bidsCheckVoxelSize(*opt*)

Check that all file to preprocess have the same voxel size

USAGE:

```
status = bidsCheckVoxelSize(opt)
```

Parameters

opt (structure) – Options chosen for the analysis. See checkOptions.

bidsCopyInputFolder(*varargin*)

Copies data from the `opt.dir.input` folder to the `opt.dir.output` folder

Then it will search the derivatives directory for any zipped `*.gz` image and uncompress the files of interest.

USAGE:

```
bidsCopyInputFolder(opt, 'unzip', true, 'force', true)
```

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **unzip** (boolean) – defaults to true
- **force** (boolean) – defaults to true

See also: `bids.copy_to_derivative`

bidsInverseNormalize(*opt*)

Brief workflow description

USAGE:

```
matlabbatch = bidsInverseNormalize(opt)
```

Parameters

opt (structure) – structure or json filename containing the options. See

bidsQApreproc(*opt*)

USAGE:

```
bidsQApreproc(opt)
```

Parameters
opt (structure) – Options chosen for the analysis. See checkOptions.

For each run works on the realigned (and unwarped) data:

- plots motion, global signal, framewise displacement

bidsRename(*opt*)

Renames SPM output into BIDS compatible files.

USAGE:

```
bidsRename(opt)
```

Parameters
opt (structure) – Options chosen for the analysis. See checkOptions.

See the spm_2_bids submodule and `defaults.set_spm_2_bids_defaults` for more info.

bidsReport(*opt*)

Prints out a human readable description of a BIDS data set for every subject in `opt.subjects`.

The output is a markdown file save in the directory:

```
opt.dir.output, 'reports', ['sub-' subLabel]
```

USAGE:

```
opt.dir.input = "path_to_dataset"
bidsReport(opt)
```

Parameters
opt (structure) – Options chosen for the analysis. See checkOptions.

20.2 workflows roi

bidsCreateROI(*opt*)

Use CPP_ROI and marsbar to create a ROI in MNI space based on a given atlas and inverse normalize those ROIs in native space if requested.

Parameters
opt (structure) – Options chosen for the analysis. See checkOptions.

USAGE:

```
% to create ROI in MNI space
opt.dir.roi = pwd;
opt.roi.atlas = 'wang';
opt.roi.hemi = {'L', 'R'};
opt.roi.name = {'V1v', 'V1d'};
opt.roi.space = {'IXI549Space''};

bidsCreateROI(opt);
```

(continues on next page)

(continued from previous page)

```
% to create ROI in subject space
opt.dir.roi = pwd;
opt.roi.atlas = 'wang';
opt.roi.hemi = {'L', 'R'};
opt.roi.name = {'V1v', 'V1d'};
opt.roi.space = {'IXI549Space', 'individual'};
opt.dir.input = fullfile(opt.dir.raw, '..', 'derivatives', 'bidspm-preproc');

bidsCreateROI(opt);
```

bidsRoiBasedGLM(*opt*)

Will run a GLM within a ROI using MarsBar.

USAGE:

```
bidsRoiBasedGLM(opt)
```

Parameters
opt (structure) – Options chosen for the analysis. See checkOptions.

Returns:

- skipped:

Will compute the absolute maximum percent signal change and the time course of the events or blocks of contrast specified in the BIDS model and save and plot the results in tsv / json / jpeg files.

> **Warning:** If your blocks are modelled as series of fast paced "short" events, the results of this workflow might be misleading. It might be better to make sure that the each block has a single event with a "long" duration.

Adapted from the MarsBar tutorial: lib/CPP_ROI/lib/marsbar-0.44/examples/batch

See also: bidsCreateRoi, *plotRoiTimeCourse()*, *getEventSpecificationRoiGlm()*

20.3 workflows lesion

bidsLesionAbnormalitiesDetection(*opt*, *extraOptions*)

Use the ALI toolbox to detect lesion abnormalities in anatomical image after segmentation of the image.

Requires the ALI toolbox: https://doi.org/10.3389/fnins.2013.00241

USAGE:

```
bidsLesionAbnormalitiesDetection(opt, extraOptions)
```

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **extraOptions** (structure) – Options chosen for analysis of another dataset in case they need to be merged. See checkOptions.

Lesion abnormalities detection will be performed using the information provided from the lesion segmentation output in BIDS format.

bidsLesionOverlapMap(*opt*)

Creates lesion overlap map on the anatomical image after initial segmentation and lesion abnormality detection of the image.

Requires the ALI toolbox: https://doi.org/10.3389/fnins.2013.00241

USAGE:

```
bidsLesionOverlapMap(opt)
```

Parameters
opt (`structure`) – Options chosen for the analysis. See checkOptions.

Lesion overlap map will be created using the information provided from the Lesion abnormalities detection output in BIDS format.

bidsLesionSegmentation(*opt*)

Use the ALI toolbox to perform segmentation to detect lesions of anatomical image.

Requires the ALI toolbox: https://doi.org/10.3389/fnins.2013.00241

USAGE:

```
bidsLesionSegmentation(opt)
```

Parameters
opt (`structure`) – Options chosen for the analysis. See checkOptions.

Segmentation will be performed using the information provided in the BIDS data set.

20.4 workflows stats

bidsConcatBetaTmaps(*opt*, *deleteTmaps*)

Make 4D images of beta and t-maps for the MVPA.

USAGE:

```
bidsConcatBetaTmaps(opt, deleteIndTmaps)
```

Parameters

- **opt** (`structure`) – Options chosen for the analysis. See checkOptions.
- **deleteIndTmaps** (`(boolean)`) – decide to delete t-maps. Default to `false`.

A valid BIDS stats model is required for this workflow: this is because the beta images to concatenate are those of the conditions mentioned in the `DummyContrasts` of the RUN level of the BIDS stats model.

When concatenating betamaps:

- Ensures that there is only 1 image per "contrast".
- Creates a tsv that lists the content of the 4D image.

- This TSV is in the subject level GLM folder where the beta map came from.
- This TSV file is named `sub-subLabel_task-taskName_space-space_labelfold.tsv`.

bidsFFX(*varargin*)

- specify the subject level fMRI model
- estimates it
- do both in one go
- or compute the contrasts

To run this workflows get the BOLD input images from derivatives BIDS dataset that contains the preprocessed data and get the condition, onsets, durations from the events files in the raw BIDS dataset.

For the model specification, if `opt.model.designOnly` is set to `true`, then it is possible to specify a model with no data: this can useful for debugging or to quickly inspect designs specification.

For the model estimation, it is possible to do some rough QA, by setting `opt.QA.glm.do = true`.

USAGE:

```
bidsFFX(action, opt, 'nodeName', 'run_level')
```

Parameters

action (`char`) – Action to be conducted

- `'specify'` to specify the fMRI GLM
- `'specifyAndEstimate'` for fMRI design + estimate
- `'contrasts'` to estimate contrasts.

Parameters

- **opt** (`structure`) – Options chosen for the analysis. See checkOptions.
- **nodeName** (`char`) – Only for action `'contrasts'`. Specifies which Node to work on.

See also: *setBatchSubjectLevelGLMSpec()*, *setBatchSubjectLevelContrasts()*

bidsModelSelection(*varargin*)

Uses the MACS toolbox to perform model selection.

USAGE:

```
bidsModelSelection(opt, 'action', 'all')
```

Parameters

- **opt** (`structure`) – Options chosen for the analysis. See checkOptions.
- **action** (`char`) – any of `'all'`, `'modelSpace'`, `'cvLME'`, `'posterior'`, `'BMS'`

Steps are performed in that order:

1. MA_model_space: defines a model space
2. MA_cvLME_auto: computes cross-validated log model evidence
3. MS_PPs_group_auto: calculate posterior probabilities from cvLMEs

4. MS_BMS_group_auto: perform cross-validated Bayesian model selection
5. MS_SMM_BMS: generate selected models maps from BMS

- `'all'` : performs 1 to 5
- `'modelSpace'`: : performs step 1
- `'cvLME'`: performs steps 1 and 2
- `'posterior'`: performs steps 1 and 3, assuming step 2 has already been run
- `'BMS'`: performs 1, 4 and 5, assuming step 2 and 3 have already been run

This way you can run all steps at once:

```
bidsModelSelection(opt, 'action', 'all');
```

Or in sequence (can be useful to split running cvLME in several batches of subjects)

bidsModelSelection(opt, 'action', 'cvLME'); bidsModelSelection(opt, 'action', 'posterior'); bidsModelSelection(opt, 'action', 'BMS');

Requirements:

- define the list of BIDS stats models in a cell string of fullpaths

```
opt.toolbox.MACS.model.files
```

- all models must have the same `space` and `task` defined in their inputs
- for a given subject / model, all runs must have the same numbers of regressors This requires to create dummy regressors in case some subjects are missing a condition or a confound. This can be done by using the *bidsFFX(opt)* with the option *opt.glm.useDummyRegressor* set to *true*.

Note: Adding dummy (empty) regressors will make your model non-estimable by SPM, where as the MACS toolbox can deal with this.

- specify each model for each subject:

```
opt = opt_stats_subject_level();

opt.glm.useDummyRegressor = true;

models = opt.toolbox.MACS.model.files

for i = 1:numel(models)
  opt.model.file = models{i};
  bidsFFX('specify', opt);
end
```

For more information see the toolbox manual in the folder `lib/MACS/MACS_Manual`.

Links:

- MACS toolbox repo

If you use this workflow, please cite the following paper:

```
@article{soch2018jnm,
  title={MACS - a new SPM toolbox for model assessment, comparison and selection.},
  author={Soch J, Allefeld C},
  journal={Journal of Neuroscience Methods},
  year={2018},
  volume={306},
  doi={https://doi.org/10.1016/j.jneumeth.2018.05.017}
}
```

If you use cvBMS or cvBMA, please also cite the respective method:

```
@article{soch2016nimg,
  title={How to avoid mismodelling in GLM-based fMRI data analysis:
         cross-validated Bayesian model selection.},
  author={Soch J, Haynes JD, Allefeld C},
  journal={NeuroImage},
  year={2016},
  volume={141},
  doi={https://doi.org/10.1016/j.neuroimage.2016.07.047}
}

@article{soch2017nimg,
  title={How to improve parameter estimates in GLM-based fMRI data analysis:
         cross-validated Bayesian model averaging.},
  author={Soch J, Meyer AP, Haynes JD, Allefeld C},
  journal={NeuroImage},
  year={2017},
  volume={158},
  doi={https://doi.org/10.1016/j.neuroimage.2017.06.056}
}
```

bidsRFX(*varargin*)

- creates a mean structural image and mean mask over the sample

OR

- specifies and estimates the group level model,
- computes the group level contrasts.

USAGE:

```
bidsRFX(action, opt, 'nodeName', '')
```

Parameters

- **action** (`char`) – Action to be conducted: `'RFX'` or `'meanAnatAndMask'` or `'contrast'`
- **opt** (`structure`) – Options chosen for the analysis. See checkOptions.
- **nodeName** (`char`) – name of the BIDS stats model to run analysis on

bidsResults(*varargin*)

Computes the results for a series of contrast that can be specified at the run, subject or dataset step level (see contrast specification following the BIDS stats model specification).

USAGE:

```
bidsResults(opt,'nodeName', '')
```

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions
- **nodeName** (char or cellstr) – name of the BIDS stats model Node(s) to show results of

See also: *setBatchSubjectLevelResults()*, *setBatchGroupLevelResults()*

Below is an example of how specify the option structure to get some specific results outputs for certain contrasts.

See the online documentation for example of those outputs.

The field `opt.results` allows you to get results from several Nodes from the BIDS stats model. So you could run `bidsResults` once to view results from the subject and the dataset level.

Specify a default structure result for this node:

```
opt.results(1) = defaultResultsStructure();
```

Specify the Node name (usually "run_level", "subject_levle" or "dataset_level"):

```
opt.results(1).nodeName = 'subject_level';
```

Specify the name of the contrast whose result we want to see. This must match one of the existing contrats (dummy contrast or contrast) in the BIDS stats model for that Node:

```
opt.results(1).name = 'listening_1';
```

For each contrat, you can adapt:

- voxel level threshold (`p`) [between 0 and 1]
- cluster level threshold (`k`) [positive integer]
- type of multiple comparison (`MC`):
  - `'FWE'` is the default
  - `'FDR'`
  - `'none'`

You can thus specify something different for a second contrast:

```
opt.results(2).name = {'listening_lt_baseline'};
opt.results(2).MC =  'none';
opt.results(2).p = 0.01;
opt.results(2).k = 0;
```

Specify how you want your output (all the following are on `false` by default):

```
% simple figure with glass brain view and result table
opt.results(1).png = true();

% result table as a .csv: very convenient when comes the time to write papers
opt.results(1).csv = true();
```

(continues on next page)

(continued from previous page)

```
% thresholded statistical map
opt.results(1).threshSpm = true();

% binarised thresholded statistical map (useful to create ROIs)
opt.results(1).binary = true();
```

You can also create a montage to view the results with *opt.results(1).csv = true();* activation neuroanatomical location will be labelled. You can specify the atlas to use for that by choosing between

- `'Neuromorphometrics'` (default)
- `'aal'`
- `'visfatlas'`
- `'anatomy_toobox'`
- `'hcpex'`
- `'glasser'`
- `'wang'`

```
opt.results(1).atlas = 'Neuromorphometrics';
```

on several slices at once:

```
opt.results(1).montage.do = true();

% slices position in mm [a scalar or a vector]
opt.results(1).montage.slices = -0:2:16;

% slices orientation: can be 'axial' 'sagittal' or 'coronal'
% axial is default
opt.results(1).montage.orientation = 'axial';

% path to the image to use as underlay
% Will use the SPM MNI T1 template by default
opt.results(1).montage.background = ...
     fullfile(spm('dir'), 'canonical', 'avg152T1.nii');

% Can also be a structure to pick up the correct file for each subject
% opt.results(1).montage.background = struct('suffix', 'T1w', ...
%                                            'desc', 'preproc', ...
%                                            'modality', 'anat');
```

Finally you can export as a NIDM results zip files.

NIDM results is a standardized results format that is readable by the main neuroimaging software (SPM, FSL, AFNI). Think of NIDM as BIDS for your statistical maps. One of the main other advantage is that it makes it VERY easy to share your group results on neurovault (which you should systematically do).

- NIDM paper
- NIDM specification
- *NIDM results viewer for SPM <https://github.com/incf-nidash/nidmresults-spmhtml>*

To generate NIDM results zip file for a given contrats simply:

```
opt.results(1).nidm = true();
```

bidsSmoothContrasts(*varargin*)

- smooths all contrast images created at the subject level

USAGE:

```
bidsSmoothContrasts(opt)
```

Parameters
opt (structure) – Options chosen for the analysis. See checkOptions.

20.5 workflows preproc

bidsCreateVDM(*opt*)

Creates the voxel displacement maps from the fieldmaps of a BIDS dataset.

USAGE:

```
bidsCreateVDM(opt)
```

Parameters
opt (structure) – Options chosen for the analysis. See checkOptions.

Inspired from spmup `spmup_BIDS_preprocess` (@ commit 198c980d6d7520b1a99) (URL missing)

bidsGenerateT1map(*opt*)

Brief workflow description

USAGE:

```
bidsGenerateT1map(opt)
```

Parameters
opt (structure) – Options chosen for the analysis. See checkOptions.

bidsRealignReslice(*opt*)

Realigns and reslices the functional data of a given task.

USAGE:

```
bidsRealignReslice(opt)
```

Parameters
opt (structure) – Options chosen for the analysis. See checkOptions.

Assumes that `bidsSTC()` has already been run if `opt.stc.skip` is not set to `true`.

bidsRealignUnwarp(*opt*)

Realigns and unwarps the functional data of a given task.

USAGE:

```
bidsRealignReslice(opt)
```

Parameters
opt (structure) – Options chosen for the analysis. See checkOptions.

Assumes that bidsSTC has already been run.

If the bidsCreateVDM() workflow has been run before the voxel displacement maps will be used unless opt.useFieldmaps is set to false.

bidsRemoveDummies(*varargin*)

Removes dummies from functional files

USAGE:

```
bidsRemoveDummies(opt, 'dummyScans', someInteger, 'force', false)
```

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **dummyScans** (integer >= 0) – number of volumes to remove
- **force** (boolean) – use 'force', true to remove dummy scans even if metadata say they have already been removed

EXAMPLE:

```
opt.taskName = 'auditory';
opt.dir.input = fullfile(pwd, 'inputs', 'raw');
bidsRemoveDummies(opt, 'dummyScans', 4, 'force', false);
```

bidsResliceTpmToFunc(*opt*)

Reslices the tissue probability map (TPMs) from the segmentation to the mean functional and creates a mask for the bold mean image

USAGE:

```
bidsResliceTpmToFunc(opt)
```

Parameters
opt (structure) – Options chosen for the analysis. See checkOptions.

Assumes that the anatomical has already been segmented by bidsSpatialPrepro() or bidsSegmentSkullStrip().

It is necessary to run this workflow before running the functionalQA pipeline as the computation of the tSNR by spmup requires the TPMs to have the same dimension as the functional.

bidsSTC(*opt*)

Performs the slice timing correction of the functional data.

USAGE:

```
bidsSTC(opt)
```

Parameters
opt (structure) – Options chosen for the analysis. See checkOptions.

STC will be performed using the information provided in the BIDS data set. It will use the mid-volume acquisition time point as as reference.

In general slice order and reference slice is entered in time unit (ms) (this is the BIDS way of doing things) instead of the slice index of the reference slice (the "SPM" way of doing things).

If no slice timing information is available from the file metadata this step will be skipped.

See also: *setBatchSTC()*, *getAndCheckSliceOrder()*

bidsSegmentSkullStrip(*opt*)

Segments and skullstrips the anatomical image. This workflow is already included in the bidsSpatialPrepro workflow.

USAGE:

```
bidsSegmentSkullStrip(opt)
```

Parameters
opt (structure) – Options chosen for the analysis. See checkOptions.

bidsSmoothing(*opt*)

This performs smoothing to the functional data using a full width half maximum smoothing kernel of size "opt.fwhm.func".

USAGE:

```
bidsSmoothing(opt)
```

Parameters
opt (structure) – Options chosen for the analysis. See checkOptions.

bidsSpatialPrepro(*opt*)

Performs spatial preprocessing of the functional and anatomical data.

The anatomical data are segmented, skull-stripped [and normalized to MNI space].

The functional data are re-aligned (unwarped), coregistered with the anatomical, [and normalized to MNI space].

Assumes that `bidsSTC()` has already been run if `opt.stc.skip` is not set to `true`.

USAGE:

```
bidsSpatialPrepro([opt])
```

Parameters
opt (structure) – Options chosen for the analysis. See checkOptions.

If you want to:

- only do realign and not realign AND unwarp, make sure you set `opt.realign.useUnwarp` to `false`.

- normalize the data to MNI space, make sure `opt.space` includes `IXI549Space`.

See the *preprocessing* section of the FAQ to know at what resolution files are resampled during normalization.

If you want to:

- use another type of anatomical data than `T1w` as a reference or want to specify which anatomical session is to be used as a reference, you can set this in `opt.bidsFilterFiler.t1w`:

```
opt.bidsFilterFiler.t1w.suffix = 'T1w';
opt.bidsFilterFiler.t1w.ses = 1;
```

bidsWholeBrainFuncMask(*opt*)

Create segmented-skull stripped mean functional image

20.6 batches

saveMatlabBatch(*matlabbatch*, *batchType*, *opt*, *subLabel*)

#ok<INUSL>

Saves the matlabbatch job in a .m file. Environment information are saved in a .json file.

% USAGE:

```
saveMatlabBatch(matlabbatch, batchType, opt, [subLabel])
```

Parameters

- **matlabbatch** (`structure`)
- **batchType** (`char`)
- **opt** (`structure`) – Options chosen for the analysis. See checkOptions.
- **subLabel** (`char`)

The .m file can directly be loaded with the SPM batch or run directly by SPM standalone or SPM docker.

The .json file also contains heaps of info about the "environment" used to set up that batch including the version of:

- OS,
- MATLAB or Octave,
- SPM,
- bidspm

This can be useful for methods writing though if the the batch is generated in one environment and run in another (for example set up the batch with Octave on Mac OS and run the batch with Docker SPM), then this information will be of little value in terms of computational reproducibility.

setBachRename(*varargin*)

USAGE:

```
matlabbatch = setBachRename(matlabbatch, files, moveTo, patternReplace,␣
↪overwriteDuplicate)
```

Returns

- **matlabbatch**
  (cell) The matlabbatch ready to run the spm job

setBatch3Dto4D(*matlabbatch*, *opt*, *volumesList*, *RT*, *outputName*, *dataType*)

Set the batch for 3D to 4D conversion

USAGE:

```
matlabbatch = setBatch3Dto4D(matlabbatch, volumesList, RT, [outputName], [dataType])
```

Parameters

- **matlabbatch** (structure)
- **volumesList** (array) – List of volumes to be converted in a single 4D brain
- **outputName** (char) – The string that will be used to save the 4D brain
- **dataType** (integer) – It identifies the data format conversion
- **RT** (float) – It identifies the TR in seconds of the volumes to be written in the 4D file header

Returns

- **matlabbatch**
  (structure) The matlabbatch ready to run the spm job

dataType:

- 0: SAME
- 2: UINT8 - unsigned char
- 4: INT16 - signed short
- 8: INT32 - signed int
- 16: FLOAT32 - single prec. float
- 64: FLOAT64 - double prec. float

setBatchImageCalculation(*varargin*)

Set a batch for a image calculation

USAGE:

```
matlabbatch = setBatchImageCalculation(matlabbatch, input, output, outDir,␣
↪expression)
```

Parameters

- **matlabbatch** (structure)
- **input** (cell) – list of images
- **output** (char) – name of the output file
- **outDir** (char) – output directory
- **expression** (char) – mathematical expression to apply (for example '(i1+i2)>3')

- **expression** – data type that must be one of the following: - ‘uint8’ - ‘int16’ (default) - ‘int32’ - ‘float32’ - ‘float64’ - ‘int8’ - ‘uint16’ - ‘uint32’

See `spm_cfg_imcalc.m` for more information:

```
``edit(fullfile(spm('dir'), 'config', 'spm_cfg_imcalc.m'))``
```

Returns

- **matlabbatch**

setBatchMeanAnatAndMask(*matlabbatch*, *opt*, *outputDir*)

Creates batxh to create mean anatomical image and a group mask

USAGE:

```
matlabbatch = setBatchMeanAnatAndMask(matlabbatch, opt, funcFWHM, outputDir)
```

Parameters

- **matlabbatch** (structure)
- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **outputDir** (string)

Returns

- **matlabbatch**
  (structure)

setBatchPrintFigure(*matlabbatch*, *opt*, *figureName*)

template to creae new setBatch functions

USAGE:

```
matlabbatch = setBatchPrintFigure(matlabbatch, figureName)
```

Parameters

- **matlabbatch**
- **figureName** (string)

Returns

- **matlabbatch**
  (structure) The matlabbatch ready to run the spm job

setBatchReorient(*varargin*)

Set up a batch to reorient images.

USAGE:

```
matlabbatch = setBatchReorient(matlabbatch, opt, images, reorientMatrix, 'prefix', '
↪')
```

Parameters

- **matlabbatch** (cell) – matlabbatch to append to.
- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **images** (cell string)
- **reorientMatrix** – 4 X 4 transformation matrix or .mat file containing a transformationMatrix variable
- **prefix** (char)

Returns

- **matlabbatch**
  (cell) The matlabbatch ready to run the spm job

setBatchSelectAnat(*matlabbatch*, *BIDS*, *opt*, *subLabel*)

Creates a batch to set an anatomical image

USAGE:

```
matlabbatch = setBatchSelectAnat(matlabbatch, BIDS, opt, subLabel)
```

Parameters

- **matlabbatch** (structure) – list of SPM batches
- **BIDS** (structure) – dataset layout.

See also: bids.layout, *getData()*.

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **subLabel** (char) – subject label

Returns

matlabbatch
(structure)

matlabbatch = setBatchSelectAnat(matlabbatch, BIDS, opt, subLabel)

- image type = opt.bidsFilterFiler.t1w.suffix (default = T1w)
- session to select the anat from = opt.bidsFilterFiler.t1w.ses (default = 1)

We assume that the first anat of that type is the "correct" one

20.7 batches lesion

setBatchLesionAbnormalitiesDetection(*matlabbatch*, *opt*, *images*)

Creates a batch to detect lesion abnormalities

Requires the ALI toolbox: https://doi.org/10.3389/fnins.2013.00241

USAGE:

```
matlabbatch = setBatchLesionAbnormalitiesDetection(matlabbatch, BIDS, opt, subLabel)
```

Parameters
matlabbatch (structure) – list of SPM batches

Returns
- **matlabbatch**
  (structure)

setBatchLesionOverlapMap(*matlabbatch*, *BIDS*, *opt*, *subLabel*)

Creates a batch for the lesion overlap map

Requires the ALI toolbox: https://doi.org/10.3389/fnins.2013.00241

USAGE:

```
matlabbatch = setBatchLesionOverlapMap(matlabbatch, BIDS, opt, subLabel)
```

Parameters
matlabbatch (structure) – list of SPM batches

Returns
- **matlabbatch**
  (structure)

setBatchLesionSegmentation(*matlabbatch*, *BIDS*, *opt*, *subLabel*)

Creates a batch to segment the anatomical image for lesion detection

Requires the ALI toolbox: https://doi.org/10.3389/fnins.2013.00241

USAGE:

```
matlabbatch = setBatchSegmentationDetectLesion(matlabbatch, BIDS, opt, subLabel)
```

Parameters
matlabbatch (structure) – list of SPM batches

Returns
- **matlabbatch**
  (structure)

20.8 batches stats

setBatchContrasts(*matlabbatch*, *opt*, *spmMatFile*, *consess*)

Add a contrast to the batch for SPM.

USAGE:

```
matlabbatch = setBatchContrasts(matlabbatch, opt, spmMatFile, consess)
```

Parameters
- **matlabbatch** (cell)
- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **spmMatFile** (char)

- **consess** (cell)

Returns

- **matlabbatch**
  (structure)

setBatchEstimateModel(*matlabbatch*, *opt*, *tmp*, *contrastsList*, *groups*)

Set up the estimate model batch for run/subject or group level GLM

USAGE:

```
% for subject level
matlabbatch = setBatchEstimateModel(matlabbatch, opt, subLabel)

% for group level
matlabbatch = setBatchEstimateModel(matlabbatch, opt, nodeName, contrastsList,␣
↪groups)
```

Parameters

- **matlabbatch** (structure)
- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **nodeName** (char)
- **contrastsList** (cell string)

Returns

- **matlabbatch**
  (structure)

setBatchFactorialDesign(*matlabbatch*, *opt*, *nodeName*)

Handles group level GLM specification

USAGE:

```
[matlabbatch, contrastsList] = setBatchFactorialDesign(matlabbatch, opt, nodeName)
```

Parameters

- **matlabbatch** (structure)
- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **nodeName** (char)

Returns

- **matlabbatch**
  (structure)

setBatchFactorialDesignImplicitMasking(*factorialDesign*)

USAGE:

```
factorialDesign = setBatchFactorialDesignImplicitMasking(factorialDesign)
```

setBatchFatorialDesignGlobalCalcAndNorm(*factorialDesign*)

USAGE:

```
factorialDesign = setBatchFatorialDesignGlobalCalcAndNorm(factorialDesign)
```

setBatchGroupLevelContrasts(*matlabbatch*, *opt*, *nodeName*)

USAGE:

```
matlabbatch = setBatchGroupLevelContrasts(matlabbatch, opt, nodeName)
```

Parameters

- **matlabbatch** (structure)
- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **nodeName** (char)

Returns

- **matlabbatch**

See also: *setBatchContrasts()*, *specifyContrasts()*, *setBatchSubjectLevelContrasts()*

setBatchGroupLevelResults(*varargin*)

USAGE:

```
matlabbatch = setBatchGroupLevelResults(matlabbatch, opt, result)
```

Parameters

- **matlabbatch** (structure)
- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **result** (structure)

Returns

- **matlabbatch**
  (structure)

setBatchResults(*matlabbatch*, *opt*, *result*)

Outputs the typical matlabbatch to compute the result for a given contrast

Common for all type of results: run, session, subject, dataset

USAGE:

```
matlabbatch = setBatchResults(matlabbatch, opt, result)
```

Parameters

- **matlabbatch** (structure)
- **results**

Returns

- **matlabbatch**
  (structure)

See also: *setBatchSubjectLevelResults()*, *setBatchGroupLevelResults()*

setBatchSubjectLevelContrasts(*matlabbatch*, *opt*, *subLabel*, *nodeName*)

set batch for run and subject level contrasts

USAGE:

```
matlabbatch = setBatchSubjectLevelContrasts(matlabbatch, opt, subLabel, funcFWHM)
```

Parameters

- **matlabbatch** (structure)
- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **subLabel** (char)

Returns

- **matlabbatch**

See also: bidsFFX, *specifyContrasts()*, *setBatchContrasts()*

setBatchSubjectLevelGLMSpec(*varargin*)

Sets up the subject level GLM

USAGE:

```
matlabbatch = setBatchSubjectLevelGLMSpec(matlabbatch, BIDS, opt, subLabel)
```

Parameters

- **matlabbatch** (structure)
- **BIDS** – dataset layout.

See also: bids.layout, *getData()*. type BIDS: structure

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **subLabel** (char)

Returns

- **matlabbatch**
  (structure)

setBatchSubjectLevelResults(*varargin*)

USAGE:

```
matlabbatch = setBatchSubjectLevelResults(matlabbatch, opt, subLabel, result)
```

Parameters

- **matlabbatch** (structure)
- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **subLabel** (char)

Returns

- **matlabbatch**
  (structure)

See also: *bidsResults()*, *setBatchResults()*

setBatchTwoSampleTTest(*varargin*)

Sets up a group level GLM specification for a 2 sample T test

USAGE:

```
matlabbatch = setBatchTwoSampleTTest(matlabbatch, opt, nodeName)
```

Parameters

- **matlabbatch** (cell) – matlabbatch to append to.
- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **nodeName** (char)

Returns

- **matlabbatch**
  (cell) The matlabbatch ready to run the spm job

20.9 batches preproc

setBatchComputeVDM(*matlabbatch*, *fmapType*, *refImage*)

Short description of what the function does goes here.

USAGE:

```
matlabbatch = setBatchComputeVDM(matlabbatch, fmapType, refImage)
```

Parameters

- **matlabbatch** (structure) – list of SPM batches
- **fmapType** (char) – ''phasediff'' or 'phase&mag'
- **refImage** – Reference image

Returns

- **matlabbatch**
  (structure) The matlabbatch ready to run the spm job

matlabbatch = setBatchComputeVDM(type)

adapted from spmup get_FM_workflow.m (@ commit 198c980d6d7520b1a996f0e56269e2ceab72cc83)

setBatchCoregistration(*varargin*)

Set the batch for coregistering the source images into the reference image

USAGE:

```
matlabbatch = setBatchCoregistration(matlabbatch, opt, ref, src, other)
```

Parameters

- **matlabbatch** (structure) – list of SPM batches
- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **ref** (char) – Reference image
- **src** (char) – Source image
- **other** (cell string) – Other images to apply the coregistration to

Returns

- **matlabbatch**
  (structure) The matlabbatch ready to run the spm job

setBatchCoregistrationFmap(*matlabbatch*, *BIDS*, *opt*, *subLabel*)

Set the batch for the coregistration of field maps

USAGE:

```
matlabbatch = setBatchCoregistrationFmap(matlabbatch, BIDS, opt, subLabel)
```

Parameters

BIDS (structure) – dataset layout.

See also: bids.layout, *getData()*.

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **subLabel** (char)

Returns

- **matlabbatch**
  (structure) The matlabbatch ready to run the spm job

TODO implement for 'phase12', 'fieldmap', 'epi'

setBatchCoregistrationFuncToAnat(*matlabbatch*, *BIDS*, *opt*, *subLabel*)

Set the batch for coregistering the functional images to the anatomical image.

USAGE:

```
matlabbatch = setBatchCoregistrationFuncToAnat(matlabbatch, BIDS, subLabel, opt)
```

Parameters

- **matlabbatch** (structure) – list of SPM batches
- **BIDS** (structure) – dataset layout.

See also: bids.layout, *getData()*

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **subLabel** (char) – subject label

Returns

- **matlabbatch**
  (structure) The matlabbatch ready to run the spm job

setBatchCreateVDMs(*matlabbatch*, *BIDS*, *opt*, *subLabel*)

Short description of what the function does goes here.

USAGE:

```
matlabbatch = setBatchCreateVDMs(matlabbatch, BIDS, opt, subLabel)
```

Parameters

- **matlabbatch** (structure)
- **BIDS** (structure) – dataset layout.

See also: bids.layout, *getData()*.

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **subLabel** (char) – subject label

Returns

- **matlabbatch**
  (structure) The matlabbatch ready to run the spm job

TODO implement for 'phase12', 'fieldmap', 'epi'

setBatchGenerateT1map(*varargin*)

batch to create a T1 and R1 map from MP2RAGE

USAGE:

```
matlabbatch = setBatchGenerateT1map(matlabbatch, BIDS, opt, subLabel)
```

Parameters

- **matlabbatch** (cell) – matlabbatch to append to.
- **BIDS** – dataset layout.

See also: bids.layout, *getData()*. type BIDS: structure

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **subLabel** (char) – subject label

Returns

- **matlabbatch**
  (cell) The matlabbatch ready to run the spm job

Relies on the MP2RAGE toolbox for SPM.

https://github.com/benoitberanger/mp2rage

Some non-BIDS metadata need to be added to the JSON files of the inversion images for this to work (check the README of the toolbox for more info):

- `EchoSpacing`
- `SlicePartialFourier` or `PartialFourierInSlice`: between 0 and 1 (example: 6/8)
- `FatSat`: must be "yes" or "no"

Most of the those metadata should be available from the PDF of with yout sequence details.

setBatchInverseNormalize(*matlabbatch*, *BIDS*, *opt*, *subLabel*, *imgToResample*)

Short description of what the function does goes here.

USAGE:

```
matlabbatch = setBatchNormalize(matlabbatch, deformField, subLabel, imgToResample)
```

Parameters
matlabbatch (`structure`)

Returns

- **matlabbatch**
  (structure)

setBatchNormalizationSpatialPrepro(*matlabbatch*, *BIDS*, *opt*, *voxDim*)

Short description of what the function does goes here.

USAGE:

```
matlabbatch = setBatchNormalizationSpatialPrepro(matlabbatch, opt, voxDim)
```

Parameters

- **matlabbatch** (`structure`)
- **opt** (`array`) – Options chosen for the analysis. See checkOptions.
- **voxDim**

Returns

- **matlabbatch**
  (structure)

setBatchNormalize(*matlabbatch*, *deformField*, *voxDim*, *imgToResample*)

Short description of what the function does goes here.

USAGE:

```
matlabbatch = setBatchNormalize(matlabbatch [, deformField] [, voxDim] [,␣
↪imgToResample])
```

Parameters

- **matlabbatch** (`structure`)
- **deformField**
- **voxDim**
- **imgToResample**

Returns

- **matlabbatch**
  (structure)

setBatchRealign(*varargin*)

Set the batch for realign / realign and reslice / realign and unwarp

USAGE:

```
[matlabbatch, voxDim] = setBatchRealign(matlabbatch, ...
                                        BIDS, ...
                                        opt, ...
                                        subLabel, ...
                                        [action = 'realign'])
```

Parameters

- **matlabbatch** (cell) – SPM batch
- **BIDS** (structure) – dataset layout.

See also: bids.layout, *getData()*.

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **subLabel** (char) – subject label
- **action** (char) – realign, realignReslice, realignUnwarp, 'reslice'

Returns

- **matlabbatch**
  (structure)
- **voxDim**
  (array)
- **srcMetadata**
  (structure)

setBatchRenameSegmentParameter(*varargin*)

USAGE:

```
matlabbatch = setBachRenameSegmentParameter(matlabbatch, opt)
```

Parameters

- **matlabbatch** (cell) – matlabbatch to append to.
- **opt** (structure) – Options chosen for the analysis. See checkOptions.

Returns

- **matlabbatch**
  (cell) The matlabbatch ready to run the spm job

setBatchReslice(*matlabbatch*, *opt*, *referenceImg*, *sourceImages*, *interp*)

Set the batch for reslicing source images to the reference image resolution

USAGE:

```
matlabbatch = setBatchReslice(matlabbatch, opt, referenceImg, sourceImages, interp)
```

Parameters

- **matlabbatch** (structure) – list of SPM batches
- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **referenceImg** (char or cellstring) – Reference image (only one image)
- **sourceImages** (char or cellstring) – Source images
- **interp** (integer >= 0) – type of interpolation to use (default = 4). Nearest neighbour = 0.

Returns

- **matlabbatch**
  (structure) The matlabbatch ready to run the spm job

setBatchSTC(*varargin*)

Creates batch for slice timing correction

USAGE:

```
matlabbatch = setBatchSTC(matlabbatch, BIDS, opt, subLabel)
```

Parameters

BIDS (structure) – dataset layout.

See also: bids.layout, *getData()*.

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **subLabel** (char) – subject label

Returns

- **matlabbatch**
  (structure) The matlabbatch ready to run the spm job

Slice timing units is in seconds to be BIDS compliant and not in slice number as is more traditionally the case with SPM.

If no slice order can be found, the slice timing correction will not be performed.

If not specified in the options, this function will take the mid-volume time point as reference to do the slice timing correction.

setBatchSaveCoregistrationMatrix(*matlabbatch*, *BIDS*, *opt*, *subLabel*)

Short description of what the function does goes here.

USAGE:

```
matlabbatch = setBatchSaveCoregistrationMatrix(matlabbatch, BIDS, opt, subLabel)
```

Parameters

- **matlabbatch** (structure)

- **BIDS** (structure) – dataset layout.

See also: bids.layout, *getData()*.

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **subLabel** (char)

Returns

- **matlabbatch**

setBatchSegmentation(*matlabbatch*, *opt*, *imageToSegment*)

Creates a batch to segment the anatomical image

USAGE:

```
matlabbatch = setBatchSegmentation(matlabbatch, opt)
```

Parameters

- **matlabbatch** (structure) – list of SPM batches
- **opt** (structure) – Options chosen for the analysis. See checkOptions.

Returns

matlabbatch
(structure)

setBatchSkullStripping(*matlabbatch*, *BIDS*, *opt*, *subLabel*)

Creates a batch to compute a brain mask based on the tissue probability maps from the segmentation.

USAGE:

```
matlabbatch = setBatchSkullStripping(matlabbatch, BIDS, opt, subLabel)
```

Parameters

- **matlabbatch** (structure) – list of SPM batches
- **BIDS** (structure) – dataset layout.

See also: bids.layout, *getData()*.

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **subLabel** (char) – subject label

Returns

- **matlabbatch**
(structure) The matlabbatch ready to run the spm job

This function will get its inputs from the segmentation batch by reading the dependency from opt.orderBatches.segment. If this field is not specified it will try to get the results from the segmentation by relying on the anat image returned by getAnatFilename.

The threshold for inclusion in the mask can be set by:

```
opt.skullstrip.threshold (default = 0.75)
```

Any voxel with p(grayMatter) + p(whiteMatter) + p(CSF) > threshold will be included in the skull stripping mask.

It is also possible to segment a functional image by setting `opt.skullstrip.mean` to `true`

Skullstripping can be skipped by setting `opt.skullstrip.do` to `false`

setBatchSmoothConImages(*matlabbatch*, *opt*)

Creates a batch to smooth all the con images of all subjects

USAGE:

```
matlabbatch = setBatchSmoothConImages(matlabbatch, opt)
```

Parameters

- **matlabbatch** (structure)
- **opt** (structure) – Options chosen for the analysis. See checkOptions.

Returns

- **matlabbatch**

See also: *bidsRFX()*, *setBatchSmoothing()*, *setBatchSmoothingFunc()*

setBatchSmoothing(*matlabbatch*, *opt*, *images*, *fwhm*, *prefix*)

Small wrapper to create smoothing batch

USAGE:

```
matlabbatch = setBatchSmoothing(matlabbatch, opt, images, fwhm, prefix)
```

Parameters

- **matlabbatch** (structure)
- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **images** (fullpath)
- **fwhm** (positive integer)
- **prefix** (char)

Returns

- **matlabbatch**
  (structure)

See also: `bidsSmoothing`, *bidsRFX()*, *setBatchSmoothingFunc()*, *setBatchSmoothConImages()*

setBatchSmoothingAnat(*matlabbatch*, *BIDS*, *opt*, *subLabel*)

Creates a batch to smooth the anat files of a subject

USAGE:

```
matlabbatch = setBatchSmoothingAnat(matlabbatch, BIDS, opt, subLabel)
```

Parameters

- **matlabbatch** (structure)
- **BIDS** (structure) – dataset layout.
- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **subLabel**

Returns

- **matlabbatch**
  (structure)

See also: bidsSmoothing, *setBatchSmoothing()*

setBatchSmoothingFunc(*matlabbatch*, *BIDS*, *opt*, *subLabel*)

Creates a batch to smooth the bold files of a subject

USAGE:

```
matlabbatch = setBatchSmoothingFunc(matlabbatch, BIDS, opt, subLabel)
```

Parameters

- **matlabbatch** (structure)
- **BIDS** (structure) – dataset layout.

See also: bids.layout, *getData()*.

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **subLabel** (char) – subject label

Returns

- **matlabbatch**
  (structure)

See also: bidsSmoothing, *setBatchSmoothing()*

20.10 IO

addGitIgnore(*fullpath*)

Adds a basic gitignore

USAGE:

```
addGitIgnore(fullpath)
```

Parameters

fullpath – char

addLicense(*fullpath*)

Copy CCO license in directory

USAGE:

```
addLicense(fullpath)
```

Parameters
fullpath – char

addReadme(*fullpath*)

Adds a basic README

USAGE:

```
addReadme(fullpath)
```

Parameters
fullpath – char

cleanCrash()

Removes any files left over from a previous unfinished run of the pipeline, like any `*.png` imgages

USAGE:

```
cleanCrash()
```

convertRealignParamToTsv(*rpTxtFile*, *opt*, *rmInput*)

Convert SPM typical realignement files to a BIDs compatible TSV one.

USAGE:

```
rpTsvFile = convertRealignParamToTsv(rpTxtFile, opt, rmInput)
```

Parameters

- **rpTxtFile** (`path`) – path to SPM realignement parameter txt file.
- **opt** (`structure`) – Options chosen for the analysis. See checkOptions.
- **rmInput** (`logical`) – Optional. Default to `false`. If true remove original txt file.

createDerivativeDir(*opt*)

Creates the derivative folder if it does not exist.

USAGE:

```
opt = createDerivativeDir(opt)
```

Parameters
opt (`structure`) – Options chosen for the analysis. See checkOptions.

`getData`(*varargin*)

Reads the specified BIDS data set and updates the list of subjects to analyze.

USAGE:

```
[BIDS, opt] = getData(opt, bidsDir, indexDependencies)
```

Parameters

- **opt** (`structure`) – Options chosen for the analysis. See checkOptions.
- **bidsDir** (`char`) – the directory where the data is ; default is : `fullfile(opt.dataDir, '..', 'derivatives', 'bidspm')`
- **indexDependencies** (`logical`) – Use ``'index_dependencies', true` in bids.layout.

Returns

- **opt**
  (structure)
- **BIDS**
  (structure)

`loadAndCheckOptions`(*optionJsonFile*)

Loads the json file provided describing the options of an analysis. It then checks its content and fills any missing fields with the defaults.

If no argument is provided, it checks in the current directory for any `opt_task-*.json` files and loads the most recent one by name (using the `date-` key).

USAGE:

```
opt = loadAndCheckOptions(optionJsonFile)
```

Parameters

optionJsonFile (`char`) – Fullpath to the json file describing the options of an analysis. It can also be an `opt` structure containing the options.

Returns

opt
(structure) Options chosen for the analysis. See `checkOptions()`.

`onsetsMatToTsv`(*varargin*)

Takes an SPM _onset.mat file and converts it to a _onsets.mat file.

Onsets are assumed to be in seconds.

USAGE:

```
onsetTsvFile = onsetMatToTsv(onsetMatFile)
```

Parameters

onsetMatFile (`fullpath`) – obligatory argument.

Returns

- **onsetTsvFile**
  (path)

overwriteDir(*directory*, *opt*)

USAGE:

```
overwriteDir(directory, opt)
```

regressorsMatToTsv(*varargin*)

Takes an SPM _desc-confounds_regressors.mat file and converts it to a _desc-confounds_regressors.tsv file.

USAGE:

```
regressorsTsvFile = regressorsMatToTsv(regressorsMatFile)
```

Parameters

regressorsMatFile (fullpath) – obligatory argument.

Returns

- **regressorsTsvFile**
  (path)

renameSegmentParameter(*BIDS*, *subLabel*, *opt*)

USAGE:

```
renameSegmentParameter(BIDS, subLabel, opt)
```

renameUnwarpParameter(*BIDS*, *subLabel*, *opt*)

USAGE:

```
renameUnwarpParameter(BIDS, subLabel, opt)
```

saveAndRunWorkflow(*matlabbatch*, *batchName*, *opt*, *subLabel*)

Saves the SPM matlabbatch and runs it

USAGE:

```
saveAndRunWorkflow(matlabbatch, batchName, opt, [subLabel])
```

Parameters

- **matlabbatch** (structure) – list of SPM batches
- **batchName** (char) – name of the batch
- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **subLabel** (char) – subject label

Return type
status

Return type
output - files generated for each batch

saveOptions(*opt*)

Saves options in a JSON file in a options folder.

USAGE:

```
saveOptions(opt)
```

Parameters
opt (structure) – Options chosen for the analysis. See checkOptions.

saveSpmScript(*varargin*)

Saves a matlabbatch as .m file

USAGE:

```
outputFilename = saveSpmScript(input, outputFilename)
```

Parameters

- **input** – a matlabbatch variable (cell) or the fullpath to a .mat file containing such matlabbatch variable.
- **outputFilename** (path) – optional. Path to output file

Returns

- **outputFilename**
  (path)

unzipAndReturnsFullpathName(*fullpathName*, *opt*)

Unzips a file if necessary

USAGE:

```
unzippedFullpathName = unzipAndReturnsFullpathName(fullpathName)
```

Parameters
fullpathName (char array)

Returns

- **unzippedFullpathName**
  (string)

20.11 QA

anatQA(*varargin*)

anatomical QA measures from the *Preprocessed Connectome Project Quality Assurance Protocol (QAP)* <http://preprocessed-connectomes-project.org/quality-assessment-protocol/>

EXAMPLE:

```
[json, fig] = anatQA(anatImageFile, grayMatterFile, whiteMatterFile, ...
                     'noBackground', true, 'visible', 'on')
```

Parameters

- **anatImageFile** (valid file path)
- **grayMatterFile** (valid file path) – gray matter probabilistic segmentation

- **whiteMatterFile** (`valid file path`) – white matter probabilistic segmentation
- **noBackground** (`logical`)
- **visible** (`char`) – figure visibility. Any of: `'off'`, `'on'`

OUTPUT

fig is a spm figure handle

json is a structure with the following fields:

- **SNR**
  [the signal-to-Noise Ratio.] That is: the mean intensity within gray and white matter divided by the standard deviation of the values outside the brain. Higher values are better.
- **CNR**
  [the Contrast to Noise Ratio.] That is: the mean of the white matter intensity values minus the mean of the gray matter intensity values divided by the standard deviation of the values outside the brain. Higher values are better.
- **FBER: Foreground to Background Energy Ratio,**
  That is: the variance of voxels in grey and white matter divided by the variance of voxels outside the brain. Higher values are better.
- **EFC**
  [Entropy Focus Criterion,] That is: the entropy of voxel intensities proportional to the maximum possible entropy for a similarly sized image. Indicates ghosting and head motion-induced blurring. Lower values are better. See <http://ieeexplore.ieee.org/document/650886/>

Note: When the background has 0 variance (e.g. a sequence with noise suppression like FLAIR) then the standard deviation of the white matter is used as reference instead of the background

GM and WM are thresholded by making them mutually exclusive. The background is found using data outside a large brain mask and trimming extreme values.

Adapted from Cyril Pernet's spmup

`censoring`(*data*)

routine that computes robust outliers for each column of the data in and return a matrix of censoring regressors (0s and a 1 per column)

EXAMPLE:

```
censoringRegressors = censoring(data)
```

INPUT data is a n volumes * m matrix to censor column wise

OUTPUT censoring_regressors matrix with ones in each column for
outliers found in coumns of data

Adapted from Cyril Pernet's spmup

`compileScrubbingStats`(*statsFolder*)

Make a list of `*_desc-confounds_timeseries.json` and compile their results in a single tsv.

EXAMPLE:

```
compileScrubbingStats(statsFolder)
```

computeDesignEfficiency(*tsvFile*, *opt*)

Calculate efficiency for fMRI GLMs. Relies on Rik Henson's fMRI_GLM_efficiency function.

For more information on design efficiency, see Jeanette Mumford excellent videos and the dedicated videos from the Principles of fMRI Part 2, Module 7-9.

> **Warning:** This function should NOT be used for proper design efficiency optimization as there are better tools for this.
>
> In general see the BrainPower doc but more specifically the tools below:
>
> - neuropower
> - some of the Canlab tools

USAGE:

```
e = computeDesignEfficiency(tsvFile, opt)
```

Parameters

- **tsvFile** (char) – Path to a bids _events.tsv file.
- **opt** (structure) – Options chosen for the analysis. See checkOptions.

Required:

- `opt.model.file`: path to bids stats model file
- `opt.TR`: inter-scan interval (s) - can be read from the `_bold.json`

Optional:

- `opt.t0`: initial transient (s) to ignore (default = 1)
- `opt.Ns`: number of scans

See also: `fMRI_GLM_efficiency`

—

EXAMPLE:

```
%% create stats model JSON
bm = BidsModel();
NodeIdx = 1;
bm.Nodes{NodeIdx}.Model.X = {'trial_type.cdt_A', 'trial_type.cdt_B'};
bm.Nodes{NodeIdx}.DummyContrasts = struct('type', 't', ...
                                          'Contrasts', {{'trial_type.cdt_A', 'trial_
↪type.cdt_B'}});

contrast = struct('type', 't', ...
                  'Name', 'A_gt_B', ...
                  'Weights', [1, -1], ...
                  'ConditionList', {{'trial_type.cdt_A', 'trial_type.cdt_B'}});

bm.Nodes{NodeIdx}.Contrasts{1} = contrast;

bm.write('smdl.json');
```

(continues on next page)

(continued from previous page)

```
%% create events TSV file
conditions = {'cdt_A', 'cdt_B'};
IBI = 5;
ISI = 0.1;
stimDuration = 1.5;
stimPerBlock = 12;
nbBlocks = 10;

trial_type = {};
onset = [];
duration = [];

time = 0;

for iBlock = 1:nbBlocks
  for cdt = 1:numel(conditions)
    for iTrial = 1:stimPerBlock
      trial_type{end + 1} = conditions{cdt}; %#ok<*SAGROW>
      onset(end + 1) = time;
      duration(end + 1) = stimDuration;
      time = time + stimDuration + ISI;
    end
    time = time + IBI;
  end
end

tsv = struct('trial_type',  {trial_type}, 'onset', onset, 'duration', duration');

bids.util.tsvwrite('events.tsv', tsv);

opt.TR = 2;

opt.model.file = fullfile(pwd, 'smdl.json');

e = computeDesignEfficiency(fullfile(pwd, 'events.tsv'), opt);
```

computeFDandRMS(*motionParameters*, *radius*)

simple routine that computes framewise displacement as a sum (FD) and RMS

Power et al. (2012) doi:10.1016/j.neuroimage.2011.10.018 Power et al. (2014) doi:10.1016/j.neuroimage.2013.08.048

Example:

```
[FD,RMS] = computeFDandRMS(motionParameters, radius)
```

Parameters

- **motion** (nX6 array) – motion parameters ([trans_x, trans_y, trans_z, .. rot_x, rot_y, rot__z])
- **radius** (nX6 array) – brain radius

See also: plotRPandFDandRMS, *getDist2surf()*

Adapted from Cyril Pernet's spmup

computeRobustOutliers(*varargin*)

Computes robust ouliers of a time series using S-outliers or Carling's k

EXAMPLE:

```
outliers = computeRobustOutliers(time_series, 'outlierType', 'Carling')
```

Parameters

- **timeSeries** (n X m array) – time_series are the time courses as column vectors
- **outlierType** (char) – any of 'S-outliers' or 'Carling'. Default to 'Carling'.

OUTPUT

Uutliers a binary vector indicating outliers.

S-outliers is the default options, it is independent of a measure of centrality as this is based on the median of pair-wise distances. This is a very sensitive measures, i.e. it has a relatively high false positive rates. As such it is a great detection tools.

The adjusted Carling's box-plot rule can also be used, and derived from the median of the data: outliers are outside the bound of median+/- k*IQR, with k = (17.63*n-23.64)/(7.74*n-3.71). This is a more specific measure, as such it is 'better' than S-outliers to regress-out, removing bad data points (assuming we don't want to 'remove' too many).

References:

- Rousseeuw, P. J., and Croux, C. (1993). Alternatives to the the median absolute deviation. J. Am. Stat. Assoc. 88, 1273-1263. <https://www.tandfonline.com/doi/abs/10.1080/01621459.1993.10476408>
- Carling, K. (2000). Resistant outlier rules and the non-Gaussian case. Stat. Data Anal. 33, 249:258. <http://www.sciencedirect.com/science/article/pii/S0167947399000572>
- Hoaglin, D.C., Iglewicz, B. (1987) Fine-tuning some resistant rules for outlier labelling. J. Amer. Statist. Assoc., 82 , 1147:1149 <http://www.tandfonline.com/doi/abs/10.1080/01621459.1986.10478363>

Adapted from Cyril Pernet's spmup

mriqcQA(*opt*, *suffix*)

uses the report from MRIQC (bold and T1) identify outliers using robust statistics (interquartile range).

https://mriqc.readthedocs.io/en/stable/iqms/bold.html https://mriqc.readthedocs.io/en/stable/iqms/t1w.html

USAGE:

```
mriqcQA(opt, suffix);
```

Parameters

opt (structure) – Options chosen for the analysis. See checkOptions.

EXAMPLE:

```
opt.dir.mriqc = '/home/remi/gin/dataset/derivatives/mriqc';

mriqcQA(opt, 'T1w');
mriqcQA(opt, 'bold');
```

Dependencies (in case you want to use it as standalone):

- bids-matlab

plotConfounds(*confounds*, *visible*)

Plots realignement parameters, framewise displacement, global signal

Parameters

- **confounds** (nXm array) – confounds
- **visible** (char) – figure visibility. Any of: 'off', 'on'

RETURNS:

- F: spm figure handle

See also: *computeFDandRMS()*

Adapted from Cyril Pernet's spmup

plotEvents(*eventsFile*, *modelFile*)

See bids.util.plot_events

plotRoiTimeCourse(*varargin*)

Plots the peristimulus histogram from a ROI based GLM

USAGE:

```
plotRoiTimeCourse(tsvFile, verbose, 'colors', colors, 'roiName', roiName)
```

Parameters

- **tsvFile** (path) – Content of TSV is organized in a "BIDS" way. Must be (t + 1) X c with t = time points and c = conditions. The + 1 for the row dimension is because of the headers giving the name of the condition. A side car JSON is expected to contain a SamplingFrequency field for the temporal resolution.
- **verbose** (boolean) – to show figure or not
- **colors** (n X 3 array)
- **roiName** (char)

See also: *bidsRoiBasedGLM()*

realignQA(*varargin*)

Implement different quality control fMRI realigned.

USAGE:

```
[confoundsTsv, figureHandle] = realignQA(boldFile, motionFile, ...
                                         'radius', 50, ...
                                         'visible', 'on')
```

Adapted from Cyril Pernet's spmup

20.12 bids

addStcToQuery(*BIDS*, *opt*, *subLabel*)

USAGE:

```
opt = addStcToQuery(opt, subLabel)
```

In case slice timing correction was performed this update the query to fetch the correct files for realignment.

buildIndividualSpaceRoiFilename(*deformationField*, *roiFilename*)

Creates a roi filename in individual space given a roi filename and a deformation field

USAGE:

```
roiBidsFile = buildIndividualSpaceRoiFilename(deformationField, roiFilename)
```

Parameters

- **deformationField** (path) – path to deformation field image.
- **roiFilename** (path) – path to roi image.

Returns

- **roiBidsFile**
  (bids.File object)

checkColumnParticipantsTsv(*BIDS*, *columnHdr*)

Check that a given column exists in participants.tsv

USAGE:

```
checkColumnParticipantsTsv(BIDS, columnHdr)
```

checkFmriprep(*BIDS*)

only fmriprep version >= 1.2 supported

USAGE:

```
status = checkFmriprep(BIDS)
```

Parameters

BIDS – dataset layout.

See also: bids.layout, *getData()*. type BIDS: struct

fileFilterForBold(*opt*, *subLabel*, *type*)

USAGE:

```
[filter, opt] = fileFilterForBold(opt, subLabel, type)
```

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **subLabel** (char)

- **type** (char) – any of {'glm', 'stc', 'confounds', 'events'}

generatedBy(*BIDS*)

Get pipeline info

USAGE:

```
[name, version] = generatedBy(BIDS)
```

Parameters
BIDS – dataset layout.

See also: bids.layout, *getData()*. type BIDS: struct

getAnatFilename(*varargin*)

Get the filename and the directory of some anat files for a given session and run. Unzips the files if necessary.

It several images are available it will take the first one it finds.

USAGE:

```
[anatImage, anatDataDir] = getAnatFilename(BIDS, subLabel, opt, nbImgToReturn,␣
↪tolerant)
```

Parameters
BIDS – dataset layout.

See also: bids.layout, *getData()*. type BIDS: structure

Parameters

- **subLabel** (char)
- **opt** (structure) – Options chosen for the analysis. See checkOptions.

Returns

- **anatImage**
  (string)
- **anatDataDir**
  (string)

getAndCheckRepetitionTime(*varargin*)

Gets the repetition time for a given bids.query filter (for several files) Throws an error if it returns empty or finds inconsistent repetition times.

USAGE:

```
repetitionTime = getAndCheckRepetitionTime(BIDS, filter)
```

Parameters
BIDS – dataset layout.

See also: bids.layout, *getData()*. type BIDS: structure

Parameters
filter (structure) – obligatory argument.

Returns

- **repetitionTime**
  (float) (1x1)

Example:

```
filter = opt.query;
filter.sub =  subLabel;
filter.suffix = 'bold';
filter.extension = {'.nii', '.nii.gz'};
filter.prefix = '';
filter.task = opt.taskName;

TR = getAndCheckRepetitionTime(BIDS, filter);
```

getAndCheckSliceOrder(*BIDS*, *opt*, *filter*)

Get the slice order information from the BIDS metadata. If inconsistent slice timing is found across files it throws an error.

USAGE:

```
sliceOrder = getAndCheckSliceOrder(opt)
```

Parameters
BIDS – dataset layout.

See also: `bids.layout`, *getData()*. `type BIDS: structure`

Parameters
opt (`structure`) – Options chosen for the analysis. See checkOptions.

Returns

- **sliceOrder**
  a vector of the time when each slice was acquired in in a volume or indicating the order of acquisition of the slices.

`getAndCheckSliceOrder` will try to read the `opt` structure for any relevant information about slice timing. If this is empty, it queries the BIDS dataset to see if there is any consistent slice timing information for a given `filter`.

See also: `bidsSTC`, *setBatchSTC()*

getBoldFilename(*varargin*)

Get the filename and the directory of a bold file for a given session / run.

Unzips the file if necessary.

USAGE:

```
[boldFilename, subFuncDataDir] = getBoldFilename(BIDS, subLabel, sessionID, runID,␣
↪opt)
```

Parameters
BIDS – dataset layout.

See also: `bids.layout`, *getData()*. `type BIDS: structure`

Parameters

- **subLabel** (char) – label of the subject ; in BIDS lingo that means that for a file name sub-02_task-foo_bold.nii the subLabel will be the string 02
- **sessionID** (char) – session label (for *ses-001*, the label will be *001*)
- **runID** (char) – run index label (for *run-001*, the label will be *001*)
- **opt** (structure) – Options chosen for the analysis. See checkOptions.

Returns

- **boldFilename**
  (string)
- **subFuncDataDir**
  (string)

getInfo(*BIDS*, *subLabel*, *opt*, *info*, *varargin*)

Wrapper function to fetch specific info in a BIDS structure returned by spm_bids.

USAGE:

```
varargout = getInfo(BIDS, subLabel, opt, info, varargin)
```

If info = sessions, this returns name of the sessions and their number:

```
[sessions, nbSessions] = getInfo(BIDS, subLabel, opt, 'sessions')
```

If info = runs, this returns name of the runs and their number for a specified session:

```
[runs, nbRuns] = getInfo(BIDS, subLabel, opt, 'runs', sessionID)
```

If info = filename, this returns the name of the file for a specified session and run:

```
filenames = getInfo(BIDS, subLabel, opt, 'filename', sessionID, runID, suffix)
```

Parameters

BIDS – dataset layout.

See also: bids.layout, *getData()*. type BIDS: structure

Parameters

- **subLabel** (char) – label of the subject ; in BIDS lingo that means that for a file name sub-02_task-foo_bold.nii the subLabel will be the string 02
- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **info** (char) – sessions, runs, filename, metadata
- **sessionLabel** (char) – session label (for *ses-001*, the label will be *001*)
- **runIdx** (char) – run index label (for *run-001*, the label will be *001*)
- **suffix** (char) – datatype (bold, events, physio)

getMeanFuncFilename(*BIDS*, *subLabel*, *opt*)

Get the filename and the directory of an mean functional file.

USAGE:

```
[meanImage, meanFuncDir] = getMeanFuncFilename(BIDS, subLabel, opt)
```

Parameters
BIDS – dataset layout.

See also: bids.layout, *getData()*. type BIDS: structure

Parameters

- **subLabel** (char)
- **opt** (structure) – Options chosen for the analysis. See checkOptions.

Returns

- **meanImage**
  (string)
- **meanFuncDir**
  (string)

getROIs(*varargin*)

Get the rois depending on value of "opt.bidsFilterFile.roi.space":

- the group folder for space: "MNI" or "IXI549Space"
- **the sub-*/roi/sub-subLabel folder:**
  - when in individual space ('T1w')
  - or another MNI space

USAGE:

```
[roiList, roiFolder] = getROIs(opt, subLabel)
```

getSubjectList(*BIDS*, *opt*)

Returns the subjects to analyze in opt.subjects

USAGE:

```
opt = getSubjectList(BIDS, opt)
```

Parameters
BIDS – dataset layout.

See also: bids.layout, *getData()*. type BIDS: structure

Parameters
opt (structure) – Options chosen for the analysis. See checkOptions.

Returns

- **opt**
  (structure)

If no group or subject is specified in opt then all subjects are included. This is equivalent to the default:

```
opt.groups = {''};
opt.subjects = {[]};
```

If you want to run the analysis of some subjects only based on the group they belong to **as defined in the ``participants.tsv``** file, you can do it like this:

```
opt.groups = {'control'};
```

This will run the pipeline on all the `control` subjects.

If your subject label is `blnd02` (as in `sub-blnd02`) but its group affiliation in the `participants.tsv` says `control`, then this subject will NOT be included if you run the pipeline with `opt.groups = {'blnd'}`.

If you have more than 2 groups you can specify them like this:

```
opt.groups = {'cont', 'cat'};
```

You can also directly specify the subject label for the participants you want to run:

```
opt.subjects = {'01', 'cont01', 'cat02', 'ctrl02', 'blind01'};
```

And you can combine both methods:

```
opt.groups = {'blind'};
opt.subjects = {'ctrl01'};
```

This will include all `blind` subjects and `sub-ctrl01`.

getTpmFilename(*BIDS*, *anatImage*, *res*, *space*)

Gets the fullpath filenames of the tissue probability maps (TPM)

USAGE:

```
[gm, wm, csf] = getTpmFilenames(BIDS, opt, subLabel, space, res)
```

Parameters
BIDS – dataset layout.

See also: `bids.layout`, *getData()*. `type BIDS: structure`

Parameters

- **opt** (`structure`) – Options chosen for the analysis. See checkOptions.
- **anatImage**
- **anatImage** – char
- **space**
- **space** – char
- **res**
- **res** – char

Returns

- **gm**
  (string) grey matter TPM
- **wm**
  (string) white matter TPM

- **csf**
  (string) csf matter TPM

initBids(*varargin*)

Initialize a BIDS dataset and updates dataset description.

USAGE:

```
initBids(opt, 'description', '', 'force', false)
```

Parameters

opt (`structure`) – Options chosen for the analysis. See checkOptions.

removeEmptyQueryFields(*query*)

returnNameSkullstripOutput(*inputFilename*, *outputType*)

roiGlmOutputName(*opt*, *subLabel*, *roiFileName*)

(C) Copyright 2022 bidspm developers

validate(*args*)

Validate bids dataset and bids stats model

20.13 bids_model

addConfoundsToDesignMatrix(*varargin*)

Add some typical confounds to the design matrix of bids stat model.

This will update the design matrix of the root node of the model.

Similar to the module https://nilearn.github.io/dev/modules/generated/nilearn.interfaces.fmriprep.load_confounds.html

USAGE:

```
bm = addConfoundsToDesignMatrix(bm, 'strategy', strategy);
```

Parameters

- **bm** (*BidsModel* instance or path to a `_smdl.json` file) – bids stats model.
- **strategy** (`struct`) – structure describing the confoudd strategy.

  The structure must have the following field:

  – `strategy`: cell array of char with the strategies to apply.

  The structure may have the following field:

  – `motion`: motion regressors strategy

  – `scrub`: scrubbing strategy

  – `wm_csf`: white matter and cerebrospinal fluid regressors strategy

  – `non_steady_state`: non steady state regressors strategy

  See the nilearn documentation (mentioned above) for more information on the possible values those strategies can take.

- **updateName** (`logical`) – Append the name of the root node with a string describing the counfounds added.

  `rp-{motion}_scrub-{scrub}_tissue-{wm_csf}_nsso-{non_steady_state}`

  default = `false`

Return type
BidsModel instance

Returns
bids stats model with the confounds added.

EXAMPLE:

```
strategy.strategies = {'motion', 'wm_csf', 'scrub', 'non_steady_state'};
strategy.motion = 'full';
strategy.scrub = true;
strategy.non_steady_state = true;

bm = addConfoundsToDesignMatrix(path_to_statsmodel_file, 'strategy',␣
↪strategy);
```

checkContrast(*node*, *iCon*)

Validates contrast specification

put some of that in bids.Model

USAGE:

```
contrast = checkContrast(node, iCon)
```

checkGroupBy(*node*)

Only certain type of GroupBy supported for now for each level

This helps doing some defensive programming

createDefaultStatsModel(*BIDS*, *opt*, *ignore*)

Creates a default model json file for a BIDS dataset

USAGE:

```
opt = createDefaultStatsModel(BIDS, opt)
```

Parameters
BIDS – dataset layout.

See also: `bids.layout`, *getData()*. `type BIDS: struct or path`

Parameters

- **opt** (`structure`) – Options chosen for the analysis. See checkOptions.
- **ignore** (`cellstr`) – Optional. Cell string that can contain: - `"Transformations"` - `"Contrasts"` - `"Dataset"` Can be used to avoid generating certain objects of the BIDS stats model.

Returns
opt

Outputs a model file in the current directory:

```
fullfile(pwd, 'models', ['model-default' opt.taskName '_smdl.json']);
```

This model has 3 "Nodes" in that order:

- Run level:
  - will create a GLM with a design matrix that includes all all the possible type of trial_types that exist across all subjects and runs for the task specified in `opt`, as well as the realignment parameters.
  - use `DummyContrasts` to generate contrasts for each trial_type for each run. This can be useful to run MVPA analysis on the beta images of each run.
- Subject level:
  - will create a GLM with a design matrix that includes all all the possible type of trial_types that exist across all subjects and runs for the task specified in `opt`, as well as the realignment parameters.
  - use `DummyContrasts` to generate contrasts for all each trial_type across runs

  Dataset level:
  - use `DummyContrasts` to generate contrasts for each trial_type for at the group level.

EXAMPLE:

```
opt.taskName = 'myFascinatingTask';
opt.dir.raw = fullfile(pwd, 'data', 'raw');
opt = checkOptions(opt);

[BIDS, opt] = getData(opt, opt.dir.raw);

createDefaultStatsModel(BIDS, opt);
```

createModelFamilies(*varargin*)

Create a family of models from a default one.

USAGE:

```
createModelFamilies(defaultModel, multiverse, outputDir)
```

Parameters

- **defaultModel** (*BidsModel* instance or path to a `_smdl.json` file) – bids stats model that serves as template.
- **multiverse** (`struct`) – Structure to describe the multiverse of models.

  Each field of the structure is a dimension of the multiverse. Possible dimensions are:
  - `motion`: motion regressors strategy
  - `scrub`: scrubbing strategy
  - `wm_csf`: white matter and cerebrospinal fluid regressors strategy
  - `non_steady_state`: non steady state regressors strategy

EXAMPLE:

```
multiverse.motion = {'none', 'basic', 'full'};
multiverse.scrub = {false, true};
multiverse.wm_csf = {'none', 'basic', 'full'};
multiverse.non_steady_state = {false, true};

createModelFamilies(path_to_statsmodel_file, multiverse, output_path);
```

getContrastsFromParentNode(*model*, *node*)

Recursively look for contrasts at previous levels

UISAGE:

```
contrastsList = getContrastsFromParentNode(model, node)
```

getContrastsList(*model*, *node*)

Get list of names of Contrast from this Node or gets its from the previous Nodes

USAGE:

```
contrastsList = getContrastsList(model, node)
```

Parameters

- **node** (`char or structure`) – node name or node content
- **model** (`BIDS stats model object`)

Returns

contrastsList (cellstr)

getContrastsListForFactorialDesign(*opt*, *nodeName*)

assuming a GroupBy that contains at least "contrast"

we try to grab the contrasts list from the Edge.Filter otherwise we dig in this in Node or the previous one to find the list of contrasts

getDummyContrastFromParentNode(*model*, *node*)

USAGGE:

```
dummyContrastsList = getDummyContrastFromParentNode(model, node)
```

Parameters

- **node** (`struct`)
- **model** (`BidsModel object`)

getDummyContrastsList(*model*, *node*)

Get list of names of DummyContrast from this Node or gets its from the previous Nodes

USAGE:

```
dummyContrastsList = getDummyContrastsList(model, mode)
```

Parameters

- **node** (`char or structure`) – node name or node content

- **model** (BIDS stats model object)

Returns
dummyContrastsList (cellstr)

getInclusiveMask(*opt*, *nodeName*, *BIDS*, *subLabel*)

Use the mask specified in the BIDS stats model as explicit mask.

If none is specified and we are in MNI space, then use the Intra Cerebral Volume SPM mask.

USAGE:

```
mask = getInclusiveMask(opt, nodeName, BIDS, subLabel)
```

20.14 cli

baseInputParser()

Returns an input parser with the common arguments for all bids apps calls.

Type `bidspm help` for more info.

cliBayesModel(*varargin*)

Run stats on bids datasets.

Type `bidspm help` for more info.

cliCopy(*varargin*)

Copy the content of the fmriprep directory to the output directory.

Type `bidspm help` for more info.

cliCreateRoi(*varargin*)

Create a ROI from an atlas.

Type `bidspm help` for more info.

cliDefaultModel(*varargin*)

Create a default bids stats model for a raw dataset.

Type `bidspm help` for more info.

cliPreprocess(*varargin*)

Preprocess a bids dataset.

Type `bidspm help` for more info.

cliSmooth(*varargin*)

Smooth an fmriprep dataset.

Type `bidspm help` for more info.

cliStats(*varargin*)

Run stats on bids datasets.

Type `bidspm help` for more info.

getBidsFilterFile(*args*)

getOptionsFromCliArgument(*args*)

USAGE:

```
opt = getOptionsFromCliArgument(args)
```

inputParserForBayesModel()

Returns an input parser for cliSBayesModel

Type `bidspm help` for more info.

inputParserForCopy()

Returns an input parser for cliCopy.

Type `bidspm help` for more info.

inputParserForCreateModel()

Returns an input parser for cliDefaultModel.

Type `bidspm help` for more info.

inputParserForCreateRoi()

Returns an input parser for cliDefaultModel.

Type `bidspm help` for more info.

inputParserForPreprocess()

Returns an input parser for cliPreprocess.

Type `bidspm help` for more info.

inputParserForSmooth()

Returns an input parser for cliSmooth.

Type `bidspm help` for more info.

inputParserForStats()

Returns an input parser for cliStats.

Type `bidspm help` for more info.

20.15 constants

lowLevelActions()

Returns a cell array of the low level actions that can be performed

20.16 defaults

ALI_my_defaults()

Return default for the ALI toolbox.

USAGE:

```
defaults = ALI_my_defaults()
```

This is where we set the defaults we want to use for the ALI (lesion) toolbox. These will override the spm defaults. When "not enough" information is specified in the batch files, SPM falls back on the defaults to fill in the blanks. This allows to make the script simpler.

MACS_my_defaults()

USAGE:

```
defaults = MACS_my_defaults()
```

Set defaults for the MACS toolbox

checkOptions(*opt*)

Check the option inputs and add any missing field with some defaults

USAGE:

```
opt = checkOptions(opt)
```

Parameters
opt (structure) – Options chosen for the analysis.

Returns

- **opt**
  the option structure with missing values filled in by the defaults.

- **GENERIC OPTIONS**
  - opt.dir - See ``setDirectories()`.
  - opt.groups = {''} - Group of subjects to analyze
  - opt.subjects = {[]} - Suject to run in each group.
  - opt.space = {'individual', 'IXI549Space'} - Space where we conduct the analysis
  - opt.taskName
  - opt.query = struct('modality', {{'anat', 'func'}}) - a structure used to specify subset of files to only run analysis on. See bids.query() to see how to specify.

    Warning: opt.query might be progressively deprecated in favor of opt.bidsFilterFile that allows using different filters for T1w and bold data.

  - opt.bidsFilterFile - Sets how to define a typical images "bold", "T1w"... in terms of their bids entities. The default value is:

```
struct('fmap', struct('modality', 'fmap'), ...
       'bold', struct('modality', 'func', 'suffix', 'bold'), ...
       't2w',  struct('modality', 'anat', 'suffix', 'T2w'), ...
       't1w',  struct('modality', 'anat', 'space', '', 'suffix', 'T1w'),␣
↪...
       'roi',  struct('modality', 'roi', 'suffix', 'mask'), ...
       'xfm',  struct('modality', 'anat', 'suffix', 'xfm', 'to', 'T1w'));
```

  - opt.verbosity = 1 - Set it to 0 if you want to see less output on the prompt.

  - `opt.tolerant = true` - Set it to `false` if you want turn warning into errors.
  - `opt.dryRun = false` - Set it to `true` in case you don't want to run the analysis.
  - `opt.pipeline.type = 'preproc'` - Switch it to `stats` when running GLMs.
  - `opt.pipeline.name`
  - `opt.pipeline.isBms` whether this is a bayesion model selection pipeline
  - `opt.boilerplateOnly = false` - If set to `true` only creates dataset description reports and methods description. Overwrites previous versions.
  - `opt.zeropad = 2` - Number of zeros used for padding subject numbers, in case subjects should be fetched by their index `1` and not their label `01'`.
  - `opt.rename.do = true` - Set to `false` to skip renaming files with `bidsRename()`. Mostly for debugging as the output files won't be usable by any of the stats workflows.
  - `opt.rename.overwrite = true` - To overwrite any eventual previous output of `bidsRename()`.
  - `opt.msg.color = blue` - Default font color of the prompt messages.

- **PREPROCESSING OPTIONS**
  - `opt.realign.useUnwarp = true`
  - `opt.useFieldmaps = true` - When set to `true` the preprocessing pipeline will look for the voxel displacement maps (created by `bidsCreateVDM()`) and will use them for realign and unwarp.
  - `opt.fwhm.func = 6` - FWHM to apply to the preprocessed functional images.
  - `opt.anatOnly = false` - Set to `true` to only preprocess the anatomical file.
  - `opt.segment.force = false` - Set to `true` to ignore previous output of the segmentation and force to run it again
  - `opt.skullstrip.mean = false` - Set to `true` to skulstrip mean functional image
  - `opt.skullstrip.threshold = 0.75` - Threshold used for the skull stripping. Any voxel with `p(grayMatter) + p(whiteMatter) + p(CSF) > threshold` will be included in the mask.
  - `opt.skullstrip.do = true` - Set to `true` to skip skullstripping.
  - `opt.stc.skip = false` - Boolean flag to skip slice time correction or not.
  - `opt.stc.referenceSlice = []` - Reference slice (in seconds) for the slice timing correction. If left empty the mid-volume acquisition time point will be selected at run time.
  - `opt.funcVoxelDims = []` - Voxel dimensions to use for resampling of functional data at normalization.
- **STATISTICS OPTIONS**
  - `opt.model.file = ''` - Path to the BIDS model file that contains the model to specify and the contrasts to compute. A path to a dir can be passed as well. In this case all *_smdl.json files will be used and looped over. This can useful to specify several models at once Before running Bayesion model selection on them.
  - `opt.fwhm.contrast = 0` - FWHM to apply to the contrast images before bringing them at the group level.
  - `opt.model.designOnly = false` - If set to `true`, the GLM will be specified without associating any data to it. Can be useful for quick design matrix inspection before running estimation.

- `opt.glm.roibased.do = false` - Set to `true` to use the `bidsRoiBasedGLM` workflow.
- `opt.glm.useDummyRegressor = false` - Set to `true` to add dummy regressors when a condition is missing from a run. See `bidsModelSelection()` for more information.
- `opt.glm.maxNbVols = Inf` - Sets the maximum number of volumes to include in a run in a subject level GLM. This can be useful if some time series have more volumes than necessary.
- `opt.glm.keepResiduals = false` - Keep the subject level GLM residuals if set to `true`.
- `opt.glm.concatenateRuns = false` - Concatenate "vertically" the images and and onsets of several run when set to true. Necessary to run a psycho-physiologial interaction analysis.
- `opt.QA.glm.do = false` - If set to `true` the residual images of a GLM at the subject levels will be used to estimate if there is any remaining structure in the GLM residuals (the power spectra are not flat) that could indicate the subject level results are likely confounded. See `plot_power_spectra_of_GLM_residuals.m` and Accurate autocorrelation modeling substantially improves fMRI reliability for more info.

defaultContrastsStructure()

defaultResultsStructure()

getOptionsFromModel(*opt*)

USAGE:

```
opt = getOptionsFromModel(opt)
```

mniToIxi(*varargin*)

Convert mention of MNI space to the SPM default space IXI549Space

USAGE:

```
opt_out = mniToIxi(opt)
```

Parameters
foo (`structure`) – options

Returns

- **opt_out**
  (type) (structure)

setDirectories(*opt*)

USAGE:

```
opt = setDirectories(opt)
```

setRenamingConfig(*opt*, *workflowName*)

set default map for renaming for a specific workflow

USAGE:

```
opt = setRenamingConfig(opt, workflowName)
```

set_spm_2_bids_defaults(*opt*)

set default map for renaming for bidspm

USAGE:

```
opt = set_spm_2_bids_defaults(opt)
```

Further renaming mapping can then be added, changed or removed through the `opt.spm_2_bids object`.

spm_my_defaults()

USAGE:

```
spm_my_defaults()
```

This is where we set the defaults we want to use. These will override the spm defaults. When "not enough" information is specified in the batch files, SPM falls back on the defaults to fill in the blanks. This allows to make the scripts simpler.

validateResultsStructure(*result*)

make sure there is no extra field in the result structure of a bids stats model or in the

USAGE:

```
validateResultsStructure(result)
```

20.17 infra

checkDependencies(*opt*)

Checks that that the right dependencies are installeda and loads the spm defaults.

USAGE:

```
checkDependencies()
```

checkToolbox(*varargin*)

Checks that a given SPM toolbox is installed. Possible to install it if necessary.

USAGE:

```
status = checkToolbox(toolboxName, 'verbose', false, 'install', false)
```

Parameters

- **toolboxName** (char) – obligatory argument. Any of {'ALI', 'MACS', 'mp2rage'}.
- **verbose** (boolean) – parameter
- **install** (boolean) – parameter

EXAMPLE:

```
checkToolbox('MACS', 'verbose', true, 'install', true)
```

elapsedTime(*opt*, *action*, *startTime*, *runTime*, *nbIteration*)

USAGE:

```
[start, runTime] = elapsedTime(input, startTime, runTime, nbIteration)
```

getEnvInfo(*opt*)

Gets information about the environment and operating system to help generate data descriptors for the derivatives.

USAGE:

```
[OS, generatedBy] = getEnvInfo()
```

Returns

OS
(structure) (dimension)

generatedBy
(structure) (dimension)

getRepoInfo(*rootDir*)

Return the branch and commit shasum

USAGE:

```
[branch, commit] = getRepoInfo()
```

getVersion()

Reads the version number of the pipeline from the txt file in the root of the repository.

USAGE:

```
versionNumber = getVersion()
```

Returns

versionNumber
(string) Use semantic versioning format (like v0.1.0)

resizeAliMask(*opt*)

USAGE:

```
aliMask = resizeAliMask(opt)
```

returnBsmDocURL(*section*)

USAGE:

```
url = returnBsmDocURL()
```

returnHomeDir()

Return the fullpath of the home directory of the user.

USAGE:

```
home = returnHomeDir()
```

returnRepoURL()

USAGE:

```
repoURL = returnRepoURL()
```

returnRootDir()

USAGE:

```
rootDir = returnRootDir()
```

returnRtdURL(*page*, *anchor*)

USAGE:

```
rtdURL = returnRtdURL()
```

setGraphicWindow(*opt*)

Open SPM graphic window.

USAGE:

```
[interactiveWindow, graphWindow, cmdLine] = setGraphicWindow(opt)
```

Parameters

opt (structure) – Options chosen for the analysis. See checkOptions.

Returns

- **interactiveWindow**
- **graphWindow**
- **cmdLine**
  (boolean)

silenceOctaveWarning()

USAGE:

```
silenceOctaveWarning()
```

20.18 messages

bidspmHelp()

General intro function for bidspm

Note:

```
- all parameters use ``snake_case``
- most "invalid" calls simply initialize bidspm
```

BIDS APP CALLS

generic call:

```
bidspm(bids_dir, output_dir, analysis_level, ...
       'action', 'some_action', ...
       'participant_label', {}, ...
       'space', {'individual', 'IXI549Space'}, ...
       'bids_filter_file', struct([]), ...
       'verbosity', 2, ...
       'options', struct([]))
```

Obligatory parameters

Parameters

- **bids_dir** (path) – path to a raw BIDS dataset
- **output_dir** (path) – path where to output data
- **analysis_level** (string) – Can either be 'subject' or 'dataset'. Defaults to 'subject'
- **action** (char) – defines the pipeline to run; can be any of:
  - 'copy': copies fmriprep data for smoothing
  - 'preprocess': preprocesses data
  - 'smooth': smooths data
  - 'default_model': creates a default BIDS stats model
  - 'create_roi': creates ROIs from a given atlas
  - 'stats': runs model specification / estimation, contrast computation, display results
  - 'contrasts': runs contrast computation, display results
  - 'results': displays results
  - 'bms': performs bayesian model selection
  - 'specify_only' only specifies the models
- **participant_label** (cellstr) – cell of participants labels. For example: {'01', '03', '08'}. Can be a regular expression. Defaults to {}
- **space** (cell string) – Defaults to {}
- **bids_filter_file** (path) – path to JSON file or structure
- **verbosity** (positive integer) – can be any value between 0 and 3. Defaults to 2
- **options** (path to JSON file or structure) – See the checkOptions help to see the available options.

Note: Arguments passed to bidspm have priorities over the options defined in opt. For example passing the argument 'dry_run', true will override the option opt.dryRun = false.

PREPROCESSING

```
bidspm(bids_dir, output_dir, 'subject', ...
       'action', 'preprocess', ...
       'participant_label', {}, ...
       'task', '', ...
       'space', {'individual', 'IXI549Space'}, ...
       'bids_filter_file', struct([]), ...
       'verbosity', 2, ...
       'options', struct([]), ...
       'boilerplate_only', false, ...
       'dry_run', false, ...
       'dummy_scans', 0, ...
       'anat_only', false, ...
```

(continues on next page)

(continued from previous page)

```
       'ignore', {}, ...
       'fwhm', 6, ...
       'skip_validation', false)
```

Parameters

- **boilerplate_only** (logical) – Only creates dataset description reports. and methods description. Defaults to false.
- **space** (cell string) – Defaults to {}
- **task** (char) – Only a single task can be processed at once. Defaults to ''.
- **dry_run** (logical) – Defaults to false
- **dummy_scans** (positive scalar) – Number of dummy scans to remove. Defaults to 0.
- **anat_only** (logical) – Only preprocesses anatomical data. Defaults to false.
- **ignore** (cellstr) – can be any of {'fieldmaps', 'slicetiming', 'unwarp', 'qa'}
- **fwhm** (positive scalar) – Smoothing to apply to the preprocessed data. Defaults to 6.
- **skip_validation** (logical) – To skip bids dataset or bids stats model validation. Defaults to false.

COPY

Copies and unzips input data to output dir. For example for fmriprep data before smoothing.

```
bidspm(bids_dir, output_dir, 'subject', ...
       'action', 'copy', ...
       'participant_label', {}, ...
       'task', {}, ...
       'space', {'individual', 'IXI549Space'}, ...
       'bids_filter_file', struct([]), ...
       'verbosity', 2, ...
       'options', struct([]), ...
       'anat_only', false, ...
       'force', false)
```

Parameters

- **space** (cell string) – Defaults to {}
- **task** (char or cell string) – Defaults to {}
- **force** (logical) – Overwrites previous data if true. Defaults to false.
- **anat_only** (logical) – Only copies anatomical data. Defaults to false.

CREATE_ROI

Creates ROIs from a given atlas.

```
bidspm(bids_dir, output_dir, 'subject', ...
       'action', 'create_roi', ...
       'participant_label', {}, ...
```

(continues on next page)

(continued from previous page)

```
       'space', {'MNI'}, ...
       'bids_filter_file', struct([]), ...
       'preproc_dir', preproc_dir, ...
       'verbosity', 2, ...
       'options', struct([]), ...
       'roi_atlas', 'wang', ...
       'roi_name', {'V1v', 'V1d'}, ...
       'hemisphere', {'L', 'R'})
```

Parameters

- **space** (cell string) – Defaults to {}
- **roi_atlas** (char) – Can be any of:
  - 'visfatlas'
  - 'anatomy_toobox'
  - 'neuromorphometrics'
  - 'hcpex'
  - 'wang'
  - 'glasser'

  Defaults to 'neuromorphometrics'
- **roi_name** (cell string) – Name of the roi to create. If the ROI does not exist in the atlas, the list of available ROI will be returned in the error message.
- **hemisphere** (cell string containing any of 'L', 'R') – Hemisphere of the ROI to create. Not all ROIs have both hemispheres. Defaults to {'L', 'R'}.
- **preproc_dir** (path) – path to preprocessed data Necessary when using 'T1w' or 'individual' space to access deformation fields to inverse normalize ROIs.

SMOOTH:

Copies files to output dir and smooths them.

```
bidspm(bids_dir, output_dir, 'subject', ...
       'action', 'smooth', ...
       'participant_label', {}, ...
       'task', {}, ...
       'space', {'individual', 'IXI549Space'}, ...
       'bids_filter_file', struct([]), ...
       'options', struct([]), ...
       'verbosity', 2, ...
       'fwhm', 6, ...
       'dry_run', false)
```

Parameters

- **space** (cell string) – Defaults to {}
- **task** (char or cell string) – Defaults to {}
- **fwhm** (positive scalar) – Smoothing to apply to the preprocessed data. Defaults to 6.

- **dry_run** (`logical`) – Defaults to `false`.
- **anat_only** (`logical`) – Only smooths the anatomical data. Defaults to `false`.

CREATE DEFAULT_MODEL

Creates a default BIDS stats model for a given raw BIDS dataset.

```
bidspm(bids_dir, output_dir, 'dataset', ...
       'action', 'default_model', ...
       'participant_label', {}, ...
       'task', {}, ...
       'space', {'individual', 'IXI549Space'}, ...
       'bids_filter_file', struct([]), ...
       'verbosity', 2, ...
       'options', struct([]), ...
       'ignore', {})
```

Parameters

- **space** (`cell string`) – Defaults to {}
- **task** (`char or cell string`) – Defaults to {}
- **ignore** (`cell string`) – can be any of {'contrasts', 'transformations', 'dataset'}

STATS

Note:

- `'stats'` runs model specification / estimation, contrast computation, display results
- `'contrasts'` runs contrast computation, display results
- `'results'` displays results
- `'specify_only'` only specifies the models

```
bidspm(bids_dir, output_dir, 'subject', ...
       'action', 'stats', ...
       'participant_label', {}, ...
       'task', {}, ...
       'space', {'individual', 'IXI549Space'}, ...
       'bids_filter_file', struct([]), ...
       'options', struct([]), ...,
       'verbosity', 2, ...
       'preproc_dir', preproc_dir, ...
       'model_file', model_file, ...          % specific to stats
       'fwhm', 6, ...
       'dry_run', false, ...
       'boilerplate_only', false, ...
       'roi_atlas', 'neuromorphometrics', ...
       'roi_based', false, ...
       'roi_dir', '', ...
       'roi_name', {''}, ...
```

(continues on next page)

(continued from previous page)

```
       'design_only', false, ...
       'ignore', {}, ...
       'concatenate', false, ...
       'use_dummy_regressor', false)
       'skip_validation', false)
```

Obligatory parameters

Parameters

- **preproc_dir** (path) – path to preprocessed data
- **model_file** (path to JSON file or dir or structure) – Path to the BIDS model file that contains the model to specify and the contrasts to compute. A path to a dir can be passed as well. In this case all *_smdl.json files will be used and looped over. This can useful to specify several models at once Before running Bayesion model selection on them.
- **space** (cell string) – Defaults to {}

Optional parameters

Parameters

- **fwhm** (positive scalar) – smoothing level of the preprocessed data
- **design_only** (logical) – to only run the model specification
- **ignore** (cell string) – can be any of {'qa'}, to skip quality controls into a single 4D image.
- **concatenate** (logical) – will concatenate the beta images of the conditions of interest convolved by an HRF.
- **dry_run** (logical) – Defaults to false.
- **roi_atlas** (char) – Name of the atlas to use to label activations in MNI space.
- **roi_based** (logical) – Set to true to run a ROI-based analysis. Defaults to false.
- **roi_dir** (path) – Path to the directory containing the ROIs.
- **roi_name** (cell string) – Names or regex expression of the ROI to use.
- **boilerplate_only** (logical) – Only creates dataset description reports. and methods description. Defaults to false.
- **use_dummy_regressor** (logical) – If true any missing condition will be modelled by a dummy regressor of NaN. Defaults to false.

BAYESIAN MODE SELECTION

```
bidspm(bids_dir, output_dir, 'subject', ...
       'action', 'bms', ...
       'participant_label', {}, ...
       'options', struct([]), ...,
       'verbosity', 2, ...
       'models_dir', models_dir, ...
       'fwhm', 6, ...
       'dry_run', false, ...
       'skip_validation', false)
```

Parameters

- **action** (char) – Any of: {'bms', 'bms-posterior', 'bms-bms'} *'bms'* will performm all steps for the baeysian model selection. If *'bms'* has been performed than 'bms-posterior' can```'bms-bms'``` be performed one aftr the other set a new model space and perform bayesian model selection on it. See the help section of *bidsModelSelection()* for more details.
- **models_dir** – A path to a dir can be passed as well. In this case all *_smdl.json files will be used and looped over.
- **dry_run** (logical) – Defaults to false.
- **fwhm** (positive scalar) – smoothing level of the preprocessed data

---

Note: For the bayesian model selection to function you must first specify all your models using the 'specify_only' action with the options 'use_dummy_regressor', true.

```
opt.glm.useDummyRegressor = true;

bidspm(bids_dir, output_dir, 'subject', ...
      'participant_label', participant_label, ...
      'preproc_dir', preproc_dir, ...
      'action', 'specify_only', ...
      'model_file', models_dir, ...
      'use_dummy_regressor', true
      'fwhm', FWHM);
```

---

low level calls

USAGE:

```
% initialise (add relevant folders to path)
bidspm

% equivalent to
bidspm init
bidspm('action', 'init')

% help
bidspm help
bidspm('action', 'help')

% uninitialise (remove relevant folders from path)
bidspm uninit
bidspm('action', 'uninit')

% also adds folder for testing to the path
bidspm dev
bidspm('action', 'dev')

% tried to update the current branch from the upstream repository
bidspm update
bidspm('action', 'update')
```

(continues on next page)

(continued from previous page)

```
% misc
bidspm version
bidspm('action', 'version')

bidspm run_tests
bidspm('action', 'run_tests')
```

For a more readable version of this help section, see the online documentation.

bugReport(*opt*, *ME*)

Write a small bug report.

USAGE:

```
bugReport(opt)
```

deprecated(*varargin*)

Throws a deprecation warning

USAGE:

```
deprecated(functionName)
```

Parameters

functionName (path) – obligatory argument.

errorHandling(*varargin*)

USAGE:

```
errorHandling(functionName, id, msg, tolerant, verbose)
```

Parameters

- **functionName** (char)
- **id** (char) – Error or warning id
- **msg** (char) – Message to print
- **tolerant** (boolean) – If set to `true` errors are converted into warnings
- **verbose** (boolean) – If set to `0` or `false` this will silence any warning

EXAMPLE:

```
msg = sprintf('this error happened with this file %s', filename)
id = 'thisError';
errorHandling(mfilename(), id, msg, true, opt.verbosity)
```

adapted from bids-matlab

logger(*varargin*)

Returns logger message

USAGE:

```
logger(logLevel, msg, 'options', opt, 'filename', filename, 'id', id)
```

Parameters

- **logLevel** (char) – Any of {'ERROR', 'WARNING', 'INFO', 'DEBUG'}
- **msg** (char)

noRoiFound(*varargin*)

USAGE:

```
status = noRoiFound(opt, roiList, folder)
```

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **roiList** (cell) – obligatory argument.
- **folder** (path) – optional argument. default: ''

Returns

- **status**
  (boolean)

noSPMmat(*varargin*)

USAGE:

```
status = noSPMmat(opt, subLabel, spmMatFile)
```

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **subLabel** (char)
- **spmMatFile** (path)

Returns

- **status**
  (boolean)

notImplemented(*varargin*)

Throws a not implemented warning

USAGE:

```
notImplemented(functionName, msg, opt)
```

Parameters

- **functionName** (path) – obligatory argument.
- **msg** (char) – optional
- **opt** (struct)

Returns

- **status**
  (boolean)

EXAMPLE:

```
notImplemented(mfilename(), ...
               'Meaning of life the universe and everything not impemented', ...
               opt);
```

printAvailableContrasts(*varargin*)

USAGE:

```
printAvailableContrasts(SPM, opt)
```

Parameters

- **SPM** (`structure or path`) – fullpath to SPM.mat file or content of SPM.mat file
- **opt** (`structure`) – Options chosen.

See also: *returnContrastImageFile()*, *getContrastNb()*

printBatchName(*batchName*, *opt*)

printCredits(*opt*)

(C) Copyright 2019 bidspm developers

printProcessingSubject(*iSub*, *subLabel*, *opt*)

USAGE:

```
printProcessingSubject(iSub, subLabel, opt)
```

printToScreen(*varargin*)

USAGE:

```
printToScreen(msg, opt, 'format', 'blue')
```

printWorkflowName(*workflowName*, *opt*)

timeStamp()

Returns the current time in a BIDS (ish) valid format: 'yyyy-mm-ddTHH-MM'

USAGE:

```
output = timeStamp()
```

20.19 preproc fieldmaps

getBlipDirection(*metadata*)

Gets the total read out time of a sequence.

USAGE:

```
blipDir = getBlipDirection(metadata)
```

Parameters
metadata (structure) – image metadata

Returns

- **blipDir**

Used to create the voxel dsiplacement map (VDM) from the fieldmap

getMetadataFromIntendedForFunc(*BIDS*, *fmapMetadata*)

Gets metadata of the associated bold file: - finds the bold file a fmap is intended for, - parse its filename, - get its metadata.

USAGE:

```
[totalReadoutTime, blipDir] = getMetadataFromIntendedForFunc(BIDS, fmapMetadata)
```

Parameters
BIDS – dataset layout.

See also: bids.layout, *getData()*. type BIDS: structure param fmapMetadata: type fmapMetadata: structure

Returns

totalReadoutTime
(type) (dimension)

blipDir
(type) (dimension)

At the moment the VDM is created based on the characteristics of the last func file in the IntendedFor field

getTotalReadoutTime(*metadata*)

Gets the total read out time of a sequence. Used to create the voxel dsiplacement map (VDM) from the fieldmap

USAGE:

```
totalReadoutTime = getTotalReadoutTime(metadata)
```

Parameters
metadata (structure) – image metadata

Returns

- **totalReadoutTime**
(float) in millisecond

Currently this relies on the user adding extra metadata in the json of the functional files as the metadata queried are not "official" BIDS metadata but can usually be found in the DICOM headers (for example: `PixelBandwidth`)

getVdmFile(*BIDS*, *opt*, *boldFilename*)

returns the voxel displacement map associated with a given bold file

USAGE:

```
vdmFile = getVdmFile(BIDS, opt, boldFilename)
```

Parameters
BIDS (`structure`) – dataset layout.

See also: `bids.layout`, *getData()*.

Parameters

- **opt** (`structure`) – Options chosen for the analysis. See checkOptions.
- **boldFilename** (`path`)

Returns

- **vdmFile**
  (string)

20.20 preproc utils

createPialSurface(*varargin*)

creates a gifti image of the the pial surface

EXAMPLE:

```
surfaceFile = createPialSurface(grayMatterFile, whiteMatterFile, opt)
```

Parameters

- **grayMatterFile** (`valid file path`) – gray matter probabilistic segmentation
- **whiteMatterFile** (`valid file path`) – white matter probabilistic segmentation
- **opt** (`structure`)

getAcquisitionTime(*sliceOrder*, *repetitionTime*)

USAGE:

```
acquisitionTime = getAcquisitionTime(sliceOrder)
```

Parameters
metadata (`vector`) – sliceOrder

Returns

- **acquisitionTime**

removeDummies(*varargin*)

Remove dummy scans from a time series and update file metadata.

USAGE:

```
removeDummies(inputFile, dummyScans, metadata, 'force', false, 'verbose', true)
```

Parameters

- **inputFile** (structure)
- **dummyScans** (positive integer) – number of dummy scans to remove
- **metadata** (structure)
- **force** (boolean)
- **verbose** (boolean)

segmentationAlreadyDone(*anatFile*, *BIDS*)

USAGE:

```
status = checkForPreviousSegmentOutput(anatFile, BIDS, opt)
```

Parameters
anatFile (path)

Returns
returns true

to skip segmentation if done previously

skullstrippingAlreadyDone(*anatFile*, *BIDS*)

USAGE:

```
status = skullstripAlreadyDone(anatFile, BIDS, opt)
```

Parameters
anatFile (path)

Returns
returns true

to skip skullstripping if done previously

20.21 reports

boilerplate(*varargin*)

USAGE:

```
outputFile = boilerplate(opt, ...
                         'outputPath', outputPath, ...
                         'pipelineType', pipelineType, ...
                         'partialsPath', partialsPath, ...
                         'verbosity', 2)
```

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **outputPath** (char)
- **pipelineType** (char) – 'preprocess', 'stats', 'create_roi'
- **partialsPath** (path)
- **verbose** (boolean)

EXAMPLE:

```
opt.model.file = path_to_model;
opt.designType = 'event';
opt = checkOptions(opt);

outputFile = boilerplate(opt, ...
                         'outputPath', pwd, ...
                         'pipelineType', 'stats', ...
                         'verbosity', 2)
```

copyFigures(*BIDS*, *opt*, *subLabel*)

Copy the figures from spatial preprocessing into a separate folder.

USAGE:

```
copyFigures(BIDS, opt, subLabel)
```

Parameters

BIDS – dataset layout.

See also: bids.layout, *getData()*. type BIDS: structure

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **subLabel** (char) – Subject label (for example *'01'*).

copyGraphWindownOutput(*opt*, *subLabel*, *action*, *imgNb*)

Looks into the current directory for an spm_.*imgNb.png file and moves it into the output directory sub-label/figures.

The output name of the file is

yyyymmddHHMM_sub-label_task-label_action.png

USAGE:

```
imgNb = copyGraphWindownOutput(opt, subLabel, [action = '',] [imgNb = 1])
```

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **subLabel** (char) – Subject label (for example *'01'*).
- **action** (char) – Name to be given to the figure.

- **imgNb** (`vector of integer`) – Image numbers to look for. SPM increments them automatically when adding a new figure a folder.

Returns

imgNb
(integer) number of the next image to get.

20.22 stats results

convertPvalueToString(*p*)

convert p value to char

renameNidm(*opt*, *result*, *subLabel*)

Rename the nidm files to a bids friendly filename.

USAGE:

```
renameNidm(opt, result, subLabel)
```

renameOutputResults(*opt*, *result*, *subLabel*)

we create new name for the nifti output by removing the spmT_XXXX prefix and using the XXXX as label- for the file

also rename PNG and labels activations

renamePngCsvResults(*opt*, *result*, *ext*, *subLabel*)

Rename the results png and csv files to a bids friendly filename.

USAGE:

```
renamePngCsvResults(opt, result, ext, subLabel)
```

renameSpmT(*result*)

Rename the results spmT (theresholed or binarized) outputted in GLM.

USAGE:

```
renameSpmT(result)
```

returnName(*contrast*)

To help naming of files generated when computing results of a given contrast

returnResultNameSpec(*opt*, *result*)

Return a bids.File name specification for a result file

setMontage(*result*)

USAGE:

```
montage = setMontage(result)
```

setNidm(*export*, *result*)

Handles the NIDM results aspect of the result batches

USAGE:

```
export = setNidm(export, result)
```

20.23 stats subject_level

allRunsHaveSameNbRegressors(*spmMat*)

USAGE:

```
allRunsHaveSameNbRegressors(spmMatFile)
```

appendContrast(*contrasts*, *C*, *counter*, *type*)

USAGE:

```
[contrasts, counter] = appendContrast(contrasts, C, counter, type)
```

Parameters

- **contrasts** (struct)
- **C** (struct)
- **counter** (integer)
- **type** (char?)

See also: *specifyContrasts()*

checkRegressorName(*SPM*)

extra checks for `bidsModelSelection` to make sure that:

- all sessions can be vertically concatenated
- after concatenation all regressors have the same name (or that there are dummy regressors)

USAGE:

```
checkRegressorName(SPM)
```

See also: *bidsModelSelection()*

constructContrastNameFromBidsEntity(*cdtName*, *SPM*, *iSess*)

Try using bids ses and run info saved in the SPM to build a contrast name.

If no information is foun it falls back on using the the SPM session number

Parameters

- **SPM** (struct) – content of SPM.mat
- **cdtName** (char) – contrast basename
- **iSess** (int) – SPM session

convertOnsetTsvToMat(*opt*, *tsvFile*, *runDuration*, *outputDir*)

Converts an events.tsv file to an onset file suitable for SPM subject level analysis.

USAGE:

```
fullpathOnsetFilename = convertOnsetTsvToMat(opt, tsvFile)
```

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **tsvFile** (char)
- **runDuration** (numeric) – Total duration of the run (in seconds). Optional. Events occurring later than this will be excluded.
- **outputDir** (path) – Path where to save onset.mat. Optional.

Use a BIDS stats model specified in a JSON file to:

- loads events.tsv and apply the Node.Transformations to its content
- extract the trials (onsets, durations) of the conditions that should be convolved as requested from Node.Model.HRF.Variables

It then stores them in in a .mat file that can be fed directly in an SPM GLM batch as 'Multiple conditions'

Parametric modulation can be specified via columns in the TSV file starting with pmod_. These columns can be created via the use of Node.Transformations. Only polynomial 1 are supported. More complex modulation should be precomputed via the Transformations.

if opt.glm.useDummyRegressor is set to true, any missing condition will be replaced by a DummyRegressor.

Returns

fullpathOnsetFilename
(string) name of the output .mat file.

EXAMPLE:

```
tsvFile = fullfile(pwd, 'data', 'sub-03_task-VisuoTact_run-02_events.tsv');

opt.model.file = fullfile(pwd, 'models', 'model-VisuoTact_smdl.json');
opt.verbosity = 2;
opt.glm.useDummyRegressor = false;

fullpathOnsetFilename = convertOnsetTsvToMat(opt, tsvFile);
```

See also: *createAndReturnOnsetFile()*, bids.transformers

createAndReturnCounfoundMatFile(*opt*, *tsvFile*)

Creates a _regressors.mat in the subject level GLM folder.

For a given _regressors.tsv file and _model.json, it creates a _regressors.mat file that can directly be used for the GLM specification of a subject level model.

The file is moved directly into the folder of the GLM.

USAGE:

```
counfoundMatFile = createAndReturnCounfoundMatFile(opt, tsvFile)
```

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **tsvFile** (char) – fullpath name of the tsv file.

Returns

counfoundMatFile
(string) fullpath name of the file created.

See also: *setBatchSubjectLevelGLMSpec()*, *createConfounds()*

createAndReturnOnsetFile(*opt*, *subLabel*, *tsvFile*, *runDuration*)

For a given `_events.tsv` file and `_model.json`, it creates a `_onset.mat` file that can directly be used for the GLM specification of a subject level model.

The file is moved directly into the folder of the GLM.

USAGE:

```
onsetFilename = createAndReturnOnsetFile(opt, subLabel, tsvFile)
```

Parameters

- **opt** (`structure`) – Options chosen for the analysis. See checkOptions.
- **subLabel** (`char`)
- **tsvFile** (`char`) – fullpath name of the tsv file.
- **runDuration** (`numeric`) – Total duration of the run (in seconds). Optional. Events occurring later than this will be excluded.

Returns

onsetFilename
(path) fullpath name of the file created.

See also: *convertOnsetTsvToMat()*

createConfounds(*tsvContent*, *designMatrix*, *maxNbVols*)

Creates confounds to save in a mat file for easy ingestion by SPM in the subject level GLM.

Note: Any NaN value will be converted to 0.

USAGE:

```
[names, R] = createConfounds(tsvContent, designMatrix, maxNbVols)
```

Parameters

- **tsvContent** (`structure`) – output of spm_load or bids.util.tsvread
- **designMatrix** (`cell string`) – conditions included in the design matrix
- **maxNbVols** (`positive integer or Inf`) – number of volumes included in that run to limit the number of rows in the confound regressors. If `Inf` all rows will be included. Default to `Inf`.

Returns

names

Returns

R

EXAMPLE:

```
tsvFile = fullfile(some_path, 'sub-01_task-test_desc-confounds_regressors.tsv');
tsvContent = bids.util.tsvread(tsvFile);

designMatrix = {'trial_type.VisMot'
                'trial_type.VisStat'
                'trial_type.missing_condition'
                'trans_x'
                'trans_y'
                'trans_z'
                'rot_x'
                'rot_y'
                'rot_z'};

[names, R] = createConfounds(tsvContent, designMatrix, 200);

names
>>>{'trans_x'
    'trans_y'
    'trans_z'
    'rot_x'
    'rot_y'
    'rot_z'};

size(R)
>>> 200, 6
```

See also: *setBatchSubjectLevelGLMSpec()*, *createConfounds()*

deleteResidualImages(*ffxDir*)

USAGE:

```
deleteResidualImages(ffxDir)
```

Parameters
ffxDir (char)

getBoldFilenameForFFX(*varargin*)

Gets the filename for this bold run for this task for the FFX setup and check that the file with the right prefix exist

USAGE:

```
boldFilename = getBoldFilenameForFFX(BIDS, opt, subLabel, funcFWHM, iSes, iRun)
```

Parameters
BIDS – dataset layout.

See also: `bids.layout`, *getData()*. `type BIDS: structure`

Parameters

- **opt** (`structure`) – Options chosen for the analysis. See checkOptions.
- **subLabel** (`char`)

- **iSes** (integer)
- **iRun** (integer)

Returns

- **boldFilename**
  (string)

getConfoundsRegressorFilename(*BIDS*, *opt*, *subLabel*, *session*, *run*)

Gets the potential confounds files for a given subject, session, run

USAGE:

```
realignParamFile = getRealignParamFile(BIDS, subLabel, session, run, opt)
```

Parameters

BIDS – dataset layout.

See also: bids.layout, *getData()*. type BIDS: structure

Parameters

- **subLabel** (char) – label of the subject ; in BIDS lingo that means that for a file name sub-02_task-foo_bold.nii the subLabel will be the string 02
- **session** (char) – session label (for *ses-001*, the label will be *001*)
- **run** (char) – run index label (for *run-001*, the label will be *001*)
- **opt** (structure) – Options chosen for the analysis. See checkOptions.

Returns

- **filename**
  (string)

getEventSpecificationRoiGlm(*varargin*)

USAGE:

```
event_specification = getEventSpecificationRoiGlm(SPM_file, model_file)
```

Parameters

- **SPM_file** (path) – obligatory argument. fullpath to SPM.mat
- **model_file** (fullpath) – obligatory argument. fullpath to BIDS stats model

Returns

- **event_specification**
  (structure) (dimension)

event_specification(1).name 'F1' event_specification(1).event_spec [1;1] event_specification(1).duration 0

Will use the run level contrasts but falls back on the subject level, if we do not find any contrasts at the run level.

ASSUMPTION:

That all events that are "pooled" together have more or less the same duration. No check in place to warn if that is not the case.

See also: event_fitted, event_signal

getEventsData(*tsvFile*, *modelFile*)

getFFXdir(*subLabel*, *opt*)

Sets the name the FFX directory

USAGE:

```
ffxDir = getFFXdir(subLabel, opt)
```

Parameters

- **subLabel** (char)
- **opt** (structure) – Options chosen for the analysis. See checkOptions.

Returns

- **ffxDir**
  (string)

getSessionForRegressorNb(*regIdx*, *SPM*)

Use the SPM Sess index for the contrast name

USAGE:

```
iSess = getSessionForRegressorNb(regIdx, SPM)
```

newContrast(*SPM*, *conName*, *type*, *conditionList*)

Create a new contrast structure with a zero vector.

USAGE:

```
C = newContrast(SPM, conName, type, conditionList)
```

Parameters

- **SPM** (struct)
- **conName** (struct)
- **type** (char) – Contrast type. Can be 't' or 'F'.
- **conditionList**

See also: *specifyContrasts()*

orderAndPadCounfoundMatFile(*varargin*)

When doing model comparison all runs must have same number of confound regressors and have exactly the same names (be from the same conditions), so

- so we pad them with zeros if necessary
- we reorder them

USAGE:

```
status = padCounfoundMatFile(spmSess, opt)
```

Parameters

- **spmSess** (cell) – obligatory argument.

- **opt** (structure) – Options chosen for the analysis. See checkOptions.

Returns

- **status**
  (boolean)

See also: *reorderCounfounds()*

removeIntercept(*designMatrix*)

remove intercept because SPM includes it anyway

reorderCounfounds(*varargin*)

USAGE:

```
[names, R] = reorderCounfounds(names, R, allConfoundsNames)
```

Parameters

- **names** (cell) – obligatory argument.
- **R** (array) – obligatory argument.
- **allConfoundsNames** (cell) – obligatory argument.

sanitizeConfounds(*names*, *R*)

Removes columns with same content from confounds

USAGE:

```
[names, R] = sanitizeConfounds(names, R)
```

Parameters

- **names** (cell string) – name of each confound
- **R** (array) – n x m confounds matrix (n: nb timepoint, m: nb confounds)

Returns

names

Returns

R

saveRoiGlmSummaryTable(*varargin*)

Creates a single table for a subject with all ROIs and conditions

USAGE:

```
outputFile = saveRoiGlmSummaryTable(opt, subLabel, roiList, eventSpec)
```

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **subLabel** (char)
- **roiList** (cellstr) – a cellstr of roi with bids friendly filenames
- **eventSpec** (struct) – "eventSpec(iCon).name"

selectConfoundsByVarianceExplained(*tsvContent*, *metadata*, *opt*)

Selects up to X of the fmriprep counfound regressors from a specific tissue type that explain most of the variance

USAGE:

```
newTsvContent = selectConfoundsByVarianceExplained(tsvContent, metadata, opt)
```

EXAMPLE:

```
tsvContent = bids.util.tsvread(tsvFile);
metadata = bids.util.tsvread(jsonFile);

opt.columnsToSearch = {'c_comp_cor'};
opt.tissueNames = {'CSF'};
opt.maxNbRegPerTissue = 2;
opt.prefix = 'keep_';
newTsvContent = selectConfoundsByVarianceExplained(tsvContent, metadata, opt);
```

specifyContrasts(*model*, *SPM*, *nodeName*)

Specifies the contrasts for run, session and subject level nodes.

USAGE:

```
contrasts = specifyContrasts(model, SPM)
```

Parameters

- **SPM** (structure) – content of SPM.mat
- **model** (bids model object)
- **nodeName** (char) – name of the node to return name of

Returns

- **contrasts**
  (structure)

To know the names of the columns of the design matrix, type : `strvcat(SPM.xX.name)`

See also: *setBatchSubjectLevelContrasts()*, *setBatchGroupLevelContrasts()*

specifyDummyContrasts(*model*, *node*, *contrasts*, *counter*)

USAGE:

```
[contrasts, counter] = specifyDummyContrasts(model, node, contrasts, counter )
```

Parameters

- **contrasts** (struct)
- **node** (struct)
- **counter** (integer)
- **model** (BidsModel object)

See also: *specifyContrasts()*

specifySessionLvlContrasts(*model*, *node*, *contrasts*, *counter*)

USAGE:

```
[contrasts, counter] = specifySessionLvlContrasts(model, node, contrasts, counter)
```

Parameters

- **contrasts** (struct)
- **node**
- **counter** (integer)
- **model** (BidsModel object)

See also: *specifyContrasts()*

specifySubLvlContrasts(*model*, *node*, *contrasts*, *count*)

USAGE:

```
[contrasts, counter] = specifySubLvlContrasts(model, node, contrasts, counter)
```

Parameters

- **model** (BidsModel instance)
- **contrasts** (struct)
- **node** (struct)
- **counter** (integer)

See also: *specifyContrasts()*

20.24 stats group_level

computeCumulativeFwhm(*opt*)

Compute resulting fwhm when smoothing both time series and contrasts.

USAGE:

```
fwhm = computeCumulativeFwhm(opt)
```

findSubjectConImage(*varargin*)

Returns the fullpath of a con image(s) for a given subject label and contrast name(s).

USAGE:

```
file = findSubjectConImage(opt, subLabel, contrastName)
```

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **subLabel** (char)
- **contrastName** (char or cellstr)

Returns

- file : a fullpath or a cellstrng of fullpath

getRFXdir(*varargin*)

Sets the name the group level analysis directory and creates it if it does not exist

USAGE:

```
rfxDir = getRFXdir(opt, nodeName, contrastName, thisGroup)
```

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **nodeName** (char)
- **contrastName** (char)
- **thisGroup** (cellstr)

Returns

rfxDir
(string) Fullpath of the group level directory

Typical output:

- opt.dir.derivatives/bidspm-stats/derivatives/bidspm-groupStats/bidspm-stats

```
['sub-ALL-task-',     model.Input.task, ...
 '_space-'    model.Input.space, ...
 '_FWHM-',    num2str(opt.fwhm.func), ...
 '_conFWHM-', opt.fwhm.contrast, ...
 'node-', model.Input.Nodes(dataset_level).Name, ...              % optional
 'contrast-', model.Input.Nodes(dataset_level).Contrast(i).Name  % if ~= from
↪"dataset_level"
 ]
```

groupLevelGlmType(*opt*, *nodeName*)

20.25 stats utils

createGlmDirName(*opt*)

USAGE:

```
glmDirName = createGlmDirName(opt)
```

designMatrixFigureName(*varargin*)

USAGE:

```
filename = designMatrixFigureName(opt, desc, subLabel)
```

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.

- **desc** (char) – optional argument. default: ''
- **subLabel** (char) – optional argument. default: ''

Returns

- **filename**
  (string)

fillInResultStructure(*thisResult*)

Fill a structure use to display results with defaults

USAGE:

```
thisResult = fillInResultStructure(thisResult)
```

Parameters

thisResult (struct)

getContrastNb(*result*, *opt*, *SPM*)

Identify the contrast nb actually has the name the user asked

The search is regex based and any string (like 'foo') will be by default regexified (into '^foo$').

USAGE:

```
contrastNb = getContrastNb(result, opt, SPM)
```

Parameters

- **SPM** (structure or path) – content of SPM.mat file
- **result** (struct) – structure with at least a name field with a chat with the name of the contrast of interest
- **opt** (structure) – Options chosen.

See also: *printAvailableContrasts()*, *getContrastNb()*

getRegressorIdx(*cdtName*, *SPM*, *bidsSes*)

Gets from the SPM structure the regressors index corresponding to the a condition convolved with the canonical HRF. This can also look for non convolved conditions to identify a confound regressor.

Throws a warning if there is no regressor for that condition.

USAGE:

```
[cdtName, regIdx, status] = getRegressorIdx(cdtName, SPM)
```

Parameters

- **cdtName** (char or cellstr) – name of the condition to look for
- **SPM** (structure) – content of SPM.mat
- **bidsSes** (char) – bids session label

Returns

- **cdtName**
  (char) name of the condition stripped of any eventual `'trial_type.'` prefix
- **regIdx**
  (logical) vector of the columns of the design matrix containing the regressor of interest
- **status**
  (logical) is `false` if no regressor was found for that condition

labelActivations(*varargin*)

Add MNI labels to a csv output file from SPM and saves it as TSV.

Can choose which atlas to use.

USAGE:

```
tsvFile = labelActivations(csvFile, 'atlas', 'Neuromorphometrics')
```

Parameters

- **csvFile** (path)
- **atlas** (char) – Any of
  - *'Neuromorphometrics'*``
  - *'aal'*``
  - *'hcpex'*``
  - *'wang'*``
  - *'glasser'*``
  - *'visfatlas'*``

  Defaults to `'neuromorphometrics'`

Returns

- **tsvFile**
  (path)

labelSpmSessWithBidsSesAndRun(*SPM*)

Adds the bids session and run label to each SPM.Sess.

USAGE:

```
:param SPM: content of SPM.mat
:type  SPM: structure
```

returnContrastImageFile(*varargin*)

Return the contrast image file for the contrast name the user asked

The search is regex based and any string (like 'foo') will be by default regexified (into '^foo$').

USAGE:

```
conImageFile = returnContrastImageFile(SPM, name, opt)
```

Parameters

- **SPM** (`structure or path`) – fullpath to SPM.mat file or content of SPM.mat file
- **name** (`char`) – name of the contrast of interest
- **opt** (`structure`) – Options chosen.

See also: *printAvailableContrasts()*, *getContrastNb()*

returnNumberScrubbedTimePoints(*confounds*)

Return number of regressors that have one and only 1 in the whole column.

validateContrasts(*contrasts*)

USAGE:

```
validateContrasts(contrasts)
```

Parameters

contrasts (`struct`) – structure with at least fields "name", "C"

20.26 utils

checkMaskOrUnderlay(*image*, *opt*, *type*)

USAGE:

```
image = checkMaskOrUnderlay(image, opt, type)
```

Parameters

- **image** (`path`)
- **type** (`char`) – any of `{'underlay', 'background', 'mask'}`

cleanUpWorkflow(*opt*)

USAGE:

```
cleanUpWorkflow(opt)
```

computeMeanValueInMask(*image*, *mask*)

USAGE:

```
value = computeMeanValueInMask(image, mask)
```

image: image filename mask: mask filename

computeTsnr(*boldImage*)

calculate temporal SNR from single run of fMRI timeseries data

USAGE:

```
[tsnrImage, volTsnr] = computeTsnr(boldImage)
```

Parameters

boldImage (`path`) – path to the 4D nifti image. The file must have a BIDS like name (example: `key1_label1_key2-label2_suffic.nii`)

Output:

- tsnrImage: fullpath filename of the tSNR output image
- volTsnr: 3D volume of the tSNR image

Adapted from fmrwhy: https://github.com/jsheunis/fMRwhy/blob/master/fmrwhy/qc/fmrwhy_qc_calculateStats.m

Copyright 2019 Stephan Heunis

createDataDictionary(*tsvContent*)

USAGE:

```
jsonContent = createDataDictionary(tsvContent)
```

deregexify(*string*)

Removes eventual initial ^ and ending $

Input –> Output

`^foo$` –> `foo`

USAGE:

```
string = deregexify(string)
```

displayArguments(*varargin*)

(C) Copyright 2023 bidspm developers

getDist2surf(*varargin*)

Loads the pial surface and computes the mean distance to the surface. Will return a default value of 50 mm if this fails.

USAGE:

```
davg = getDist2surf(anatImage, opt)
```

Parameters

- **anatImage** (`cell`)
- **opt** (`structure`) – Options chosen for the analysis. See checkOptions.

Returns

- **davg**
  (float) (1 x 1)

Adapted from motion finger print functions and script (mw_mfp.m) from Marko Wilke https://www.medizin.uni-tuebingen.de/kinder/en/research/neuroimaging/software/ see http://www.dx.doi.org/10.1016/j.neuroimage.2011.10.043 and http://www.dx.doi.org/10.1371/journal.pone.0106498

getFuncVoxelDims(*opt*, *subFuncDataDir*, *fileName*)

Get the resolution of an image and update the relevant field in the options.

USAGE:

```
[voxDim, opt] = getFuncVoxelDims(opt, subFuncDataDir, fileName)
```

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **subFuncDataDir**
- **fileName**

Returns

- **voxDim**
- **opt**

regexify(*string*)

Turns a string into a simple regex. Useful to query bids dataset with bids.query that by default expects will treat its inputs as regex.

Input –> Output

foo –> ^foo$

USAGE:

```
string = regexify(string)
```

renamePng(*directory*, *prefix*)

Removes the _XXX suffix before the PNG extension in files generated by SPM in a directory

Will overwrite any file that already exists

USAGE:

```
renamePng(directory)
```

returnBatchFileName(*batchType*, *ext*)

USAGE:

```
batchFileName = returnBatchFileName([batchType] [, ext])
```

Parameters

- **batchType** (cell)
- **ext** (structure) – optional.

Returns

- **batchFileName**
  (path)

returnDependency(*opt*, *type*)

Use to create dependencies between batches in workflows.

USAGE:

```
dep = returnDependency(opt, type)
```

returnVolumeList(*varargin*)

USAGE:

```
volumes = returnVolumeList(opt, boldFile)
```

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **boldFile** (fullpath)

Returns

- **volumes**
  (cell string)

setFields(*structure*, *fieldsToSet*, *overwrite*)

Recursively loop through the fields of a target `structure` and sets the values as defined in the structure `fieldsToSet` if they don't exist.

Content of the target structure can be overwritten by setting the `overwrite``` to ``true`.

USAGE:

```
structure = setFields(structure, fieldsToSet, overwrite = false)
```

Parameters

- **structure**
- **fieldsToSet** (char)
- **overwrite** (boolean)

Returns

- **structure**
  (structure)

setUpWorkflow(*varargin*)

Calls some common functions to:

- check the configuration,
- remove some old files from an eventual previous crash
- loads the layout of the BIDS dataset
- tries to open a graphic window

USAGE:

```
[BIDS, opt, group] = setUpWorkflow(opt, workflowName, bidsDir, indexData)
```

Parameters

- **opt** (structure) – Options chosen for the analysis. See checkOptions.
- **workflowName** (char) – name that will be printed on screen
- **bidsDir** – optional

- **bidsDir** – fullpath, default = ‘’
- **indexData** – Set to `false` if you want to skip the datindexing with `getData`. Can be useful for some group level workflow where indexing will happen later at the batch level. This will also skip updating the subject list done by `getData`. Default to `true`.
- **indexData** – logical, default = true
- **index_dependencies** (`logical, default = true`) – Use ``’index_dependencies’, true` in bids.layout.

Returns

- **BIDS**
  (structure) returned by `getData`
- **opt**
  options checked

tempName()

Creates a temporary directory and returns its fullpath

unfoldStruct(*input*)

USAGE:

```
unfoldStruct(input)
```

validationInputFile(*dir*, *fileNamePattern*, *prefix*)

Looks for file name pattern in a given directory and returns all the files that match that pattern but throws an error if it cannot find any.

A prefix can be added to the filename.

This function is mostly used that a file exists so that an error is thrown early when building a SPM job rather than at run time.

USAGE:

```
files = validationInputFile(dir, fileName, prefix)
```

Parameters

- **dir** (`char`) – Directory where the search will be conducted.
- **fileName** (`char`) – file name pattern. Can be a regular expression except for the starting `^` and ending `$`. For example: `'sub-.*_ses-.*_task-.*_bold.nii'`.
- **prefix** (`char`) – prefix to be added to the filename pattern. This can also be a regular expression (ish). For example ,f looking for the files that start with `c1` or `c2` or `c3`, the prefix can be `c[123]`.

Returns

files
(string array) returns the fullpath file list of all the files matching the required pattern.

See also: `spm_select`

Example: % % tissueProbaMaps = validationInputFile(anatDataDir, anatImage, ‘c[12]’);

volumeSplicing(*varargin*)

Removes specific set of volumes from a nifti time series.

USAGE:

```
outputFileFullPath = volumeSplicing(inputFile, volumesToRemove)
```

Parameters

- **inputFile** (path)
- **volumesToRemove** (1xn or nX1 array)
- **outputFile** (char) – optional parameter. default: will overwrite inputFile. If only a filename is given, the file will be created in the same folder as the input file.

Returns

- **outputFileFullPath**

Example:

```
outputFileFullPath = volumeSplicing(inputFile, volumesToRemove, 'outputFile', 'foo.
↪nii.gz');
```

20.27 validators

isMni(*input*)

Check if input is a valid MNI space

USAGE:

```
[idx, allowedSpaces] = isMni(input)
```

Parameters
input (string or cell array of strings) – The space to check

Return idx
A logical array indicating which elements of input are valid MNI spaces

Rtype idx
logical array

Return allowedSpaces
A cell array of valid MNI spaces

Rtype allowedSpaces
cell array of strings

isSkullstripped(*bidsFile*)

Check if the image is skullstripped.

USAGE:

```
status = isSkullstripped(bidsFile)
```

Parameters
bidsFile (bids.File) – bids.File object

Returns
status: true if the image is skullstripped

Return type
logical

EXAMPLE:

```
bf = bids.File('sub-01_T1w', 'use_schema', false);

status = isSkullstripped(bf)
```

isTtest(*structure*)

isZipped(*file*)

USAGE:

```
status = isZipped(file)
```

CHAPTER

TWENTYONE

INDICES AND TABLES

- genindex
- search

BIBLIOGRAPHY

[OARW19] Wiktor Olszowy, John Aston, Catarina Rua, and Guy B. Williams. Accurate autocorrelation modeling substantially improves fMRI reliability. *Nature Communications*, 10(1):1220, 2019. URL: http://www.nature.com/articles/s41467-019-09230-w (visited on 2022-06-10), doi:10.1038/s41467-019-09230-w.

MATLAB MODULE INDEX

INDEX

T

U

V