SPM_batch_creator.m: Software to create and execute SPM batch jobs for statistical analyses from a json object compliant with BIDS stats model specification

Daniel Huber[a], Roberto Viviani[a,b]

[a] Institute of Psychology, University of Innsbruck, Innsbruck, Austria
[b] Psychiatry and Psychotherapy Clinic, University of Ulm, Ulm, Germany

November 2024

INSTRUCTION MANUAL

This manual covers the use of a MATLAB toolbox to conduct the statistical analysis of BIDS-compliant and non-compliant fMRI datasets with the SPM package (https://www.fil.ion.ucl.ac.uk/spm/). Our toolbox is available through a CC BY-NC license at github.com/d-ni374/SPM_batch_creator.

INTRODUCTION

The main function through which one specifies the analyses is SPM_batch_creator. This function covers the statistical analyses of pre-processed fMRI datasets, i.e., all steps bringing rawdata to statistically exploitable images have to be carried out before using the software. It is based on SPM12 (MATLAB) and includes all options offered for first and second level analyses (i.e., all options in "fMRI model specification" and "factorial design specification"). The software is designed to accept both, BIDS-compliant [1] and non-standardized (but still restricted) datasets.

The software requires a json object which is compliant with "BIDS Stats Models Specification" [2] and contains all necessary information to generate the SPM12 batch job. There are four files demonstrating the structure of this json object:

- "model-teststructure_desc-formatInfo_smdl.json": This file reproduces the structure of the object, but the values are replaced by the required input format and an explanatory text for the respective fields. It can be further seen, if a field is required or optional and if there is a default value (usually taken from SPM12). Additional fields with name-prefixes "non-field_INFO_" contain additional information for the mentioned field, but are not part of the json object itself.
- "model-teststructure_desc-validationRequirements_smdl.json": Similar to the first file, but less verbose and more focused on instructions on how the user input is validated by the program.
- "model-teststructure_desc-emptyTemplate_smdl.json": This file contains all required and optional fields and can be used to manually fill the object. Most fields are empty or 0 (in case of numbers), some contain default values or exemplary input. If completed, this file can be directly used as the input for SPM_batch_creator.
- "model-teststructure_desc-emptyTemplateFirstLv_smdl.json": Similar to the previous file, but only containing the first level node.

All of these files can be found in the sub-folder "templates".

Generally, the json object including all inputs for statistical analyses, is based on the structure given in [2]. In order to specify parameters of SPM's statistics specifications, the fields "Transformations" and "Model.Software" are used, as these offer the possibility to enter software-specific key-value pairs and thereby define all inputs needed by SPM to (1) find the files to process, (2) find required metadata, (3) assign images and metadata to the variables of the model, (4) set options for the respective model, and (5) set output and contrast options. BIDS-compliant datasets need to be organized in a folder structure shown in Fig. 1 and the files named according to entity-based conventions.

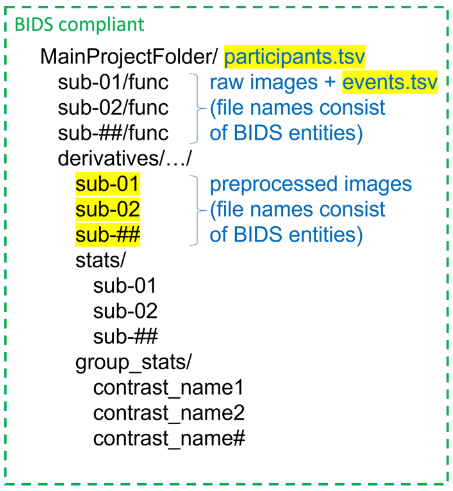


Fig. 1: Expected directory tree for BIDS-compliant datasets.

Besides the image files, events.tsv has to contain all information concerning the applied conditions (required columns: trial_type, onset, duration; optional columns for parametric modulators in the same file). If movement data or other regressors shall be included, those can be stored in a .txt, .tsv, or .mat file. For group level analysis, the file participants.tsv is required in case that covariates are intended to be included in the model (column participant_id plus one column per covariate with its name in the first row).

The workflow – starting from pre-processed MRI images and corresponding logfiles – is schematically depicted in Fig. 2. Since logfiles can have any format or structure, the user must provide events.tsv and participants.tsv in a standardized form. By help of a user interface, all required information is collected to create the BIDS-compliant json object from which first and second level batch jobs can be written and subsequently executed. Alternatively, a template of the json object can be completed manually. Additionally, a separate json file is created for each subject which contains more detailed data about the events, i.e., values like onsets, duration, etc. are stored in this file. This file is intended for quality assurance – allowing the user to control the processed data – and only supports one Run per subject/session.

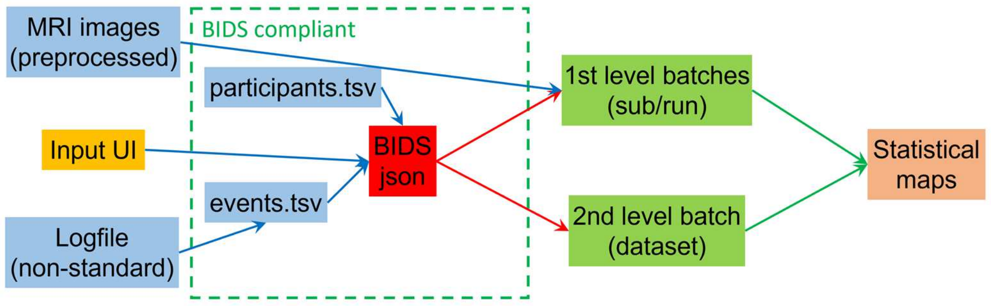
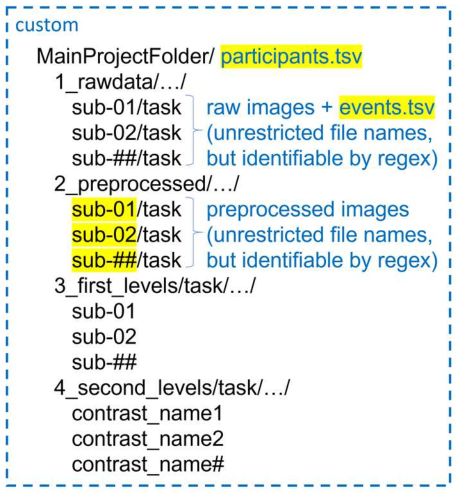


Fig. 2: Schematic representation of the analysis.

As an additional feature, the software can process datasets which do not comply with BIDS. This enables statistical analyses of datasets that were not (yet) converted to BIDS format – also including some public datasets. Because it seems impossible to cover all utilized file naming conventions and folder structures, the input data need to be distinguishable by regular expressions and saved to a certain directory tree (see Fig. 3). The names of the subfolders (“1_rawdata”, “2_preprocessed”, etc.) can be adjusted in the script *get_default_values.m*. The requirement and format of events.tsv and participants.tsv has been chosen according to the BIDS standard. In order to add flexibility to the required files, they will be searched for by regular expressions, i.e., they can be named differently, but must comply with the described structure.



```
custom
  MainProjectFolder/ participants.tsv
    1_rawdata/…/
      sub-01/task ┐ raw images + events.tsv
      sub-02/task ├ (unrestricted file names,
      sub-##/task ┘ but identifiable by regex)
    2_preprocessed/…/
      sub-01/task ┐ preprocessed images
      sub-02/task ├ (unrestricted file names,
      sub-##/task ┘ but identifiable by regex)
    3_first_levels/task/…/
      sub-01
      sub-02
      sub-##
    4_second_levels/task/…/
      contrast_name1
      contrast_name2
      contrast_name#
```

Fig. 3: Expected directory tree for non-BIDS datasets.

EXECUTION OF ANALYSES

The script SPM_batch_creator supports both, first and second level statistical analysis, and orients highly on the module layout of “fMRI model specification” and “Factorial design specification” in the SPM12 batch editor. Provided a valid json object (i.e., all required fields filled adequately), the analysis can be conducted consecutively executing:

SPM_batch_creator(1) and

SPM_batch_creator(2),

where the argument indicates the level of analysis. However, in the normal use-case the json object cannot be filled completely at once. Concerning first level analysis, contrast definition is most reliably done by directly choosing the relevant predictors from the design matrix columns and assigning them the desired weights. As the design matrix names are normally not known by the user beforehand, it is recommended to fill the json object step by step and let the script fill critical fields like Model.X which contains the list of predictors. Especially for complex designs containing several sessions, derivatives or parametric modulators, this practice shall minimize faulty inputs.

Another aspect is related to the correct definition of second level input files. SPM12 creates first level contrast files with an ascending counter (con_0001.nii, con_0002.nii, etc.), irrespective of the contrast name. More precisely, if one of the defined conditions is not met for a participant, certain contrasts cannot be created – leading to mismatching contrast enumeration among participants. Therefore, the second level analysis is based on the contrast name (which must be unique) rather than on the file name of the contrast. For this reason, the script orients itself on the contrast names stored in SPM.mat.

Since there are quite a lot of options available in the design specifications, there is an optional user interface script helping to complete the required inputs. The recommended usage of both alternatives is described below.

Notes:

- In SPM12, there is no discrimination of Sessions and Runs. Therefore, several runs per subject and multiple sessions are treated in the same way.
- The “GroupBy” functionality is not implemented yet

Part A: Script execution without using the optional user interface

Refer to the template file “model-teststructure_desc-formatInfo_smdl.json” for description of all fields and to the documentation of SPM12 (or its batch editor) for detailed explanations of the parameters. The empty model files “model-teststructure_desc-emptyTemplate_smdl.json” or “model-teststructure_desc-emptyTemplateFirstLv_smdl.json” (only first level) can be used as a starting template. These files contain some default values and some exemplary input. The bullet points in the following sections only cover fields, which are particularly important or do not directly represent an SPM12 parameter.

A.I. First level analysis

1. The first level node of the empty json template file has to be partly filled:
   - Main section: enter the model name in “Name” (this name is used for the second level sub-folder) and provide a concise “Description” (optional)
   - Entering the task name in “Input.task” is highly recommended (since this supports file filtering)
   - “Transformations.Instructions.Name”: must be “fmri_model_specification” for the first level node
   - “Input.BIDSflag”: indicate if the dataset is BIDS-compliant (1…yes, 0…no)
   - “Input.InputDirectory”: the directory directly above the sub-## folders (which contain the preprocessed images)
   - “Input.InputFilterRegexp”: regular expression to identify the preprocessed images within the folder “Input.InputDirectory”/sub-##
   - “Input.ImageType”: both, multiple 3d images or single 4d images, are supported
   - “Input.Participant_ID”: this list consists identifiers for all subjects that should be included (e.g. [“sub-01”,”sub-02”])
   - “Input.OutputDirectory”: this directory will be supplemented by a sub-folder with “Input.task” (if given) and “Input.Participant_ID”
   - “Sessions.Session_id”: can be either a number (input as e.g. “01”) or letters (in case of non-BIDS datasets, e.g. “A”); in case of non_BIDS datasets this identifier is expected to be at the end of a subfolder name within the subject’s preprocessed directory
   - “Run_ids”: Run_ids are expected to be numbers like “01”, “02”, etc. However, due to the possibility of leading zeros, they have to be entered as a list of strings; To identify runs within the folder structure, the identifier must be present in the file/folder name as “run-01”, “run_01” or “run01”
   - “EventsRegexp”: Regular expression to find events.tsv (required) for each participant, which must contain the columns “trial_type”, “onset”, and “duration”. Parametric modulators must also have their respective column in this file. events.tsv is expected to be located in the raw data folder. If it cannot be found there, it is searched for in the preprocessed folder.

- “Conditions” are defined by entering the assigned “trial_type” and choosing from the modulation options. Parametric modulators can be given a user-specified “Name”, but “ColName” must correspond to a column in events.tsv.
- “ConditionsRegexp”: Alternatively, the conditions (and its modulators) can be summarized in a .mat file -> see description in SPM’s batch editor on how the variables have to be defined. If this option is used, events.tsv is not required (in case that no “Conditions” are defined).
- “RegressorsRegexp”: The file containing regressors (e.g., movement parameters) are expected to be located in the preprocessed (or raw) data folder and are accepted as .txt, .tsv, and .mat (format requirements for .mat files can be found in the SPM batch editor).
- “FactorialDesign”: If this option is used, SPM will create automatically several contrasts related to the design (without requiring a “Contrast Manager” job). If a json file is initially created using this option, during execution of the first level jobs (by calling *SPM_batch_creator(1)*), the “FactorialDesign” section will be cleared and the associated contrasts added to the contrasts section. This step is necessary to meet BIDS specification and enables the definition of the second level inputs afterwards.
- “Model.X”: The list of regressors must agree with the column names of the design matrix. Therefore, this field is initially left empty and will be filled automatically during validation of the json file.
- “Model.HRF.Variables”: same as for “Model.X”;
- “Model.Software.SPM.Method.Type”: The option “Bayesian 1st-level” is not supported so far, but could be added in the future.
- “Contrasts” and “DummyContrasts” sections can be left empty since they rely on “Model.X”.

2. Execute *SPM_batch_creator(1)* for the first time. After validation of the json file and the given contents, the script will end with an intended error:

```
'NODE 1: The design matrix predictors are empty. Please check automatically
generated predictors and update the contrasts section with them.'
```

3. Complete the first level node by filling the “Contrasts” and “DummyContrasts” sections:
- “Name”: Each contrast must have a unique name. Since this name is also used as the folder name of the second level analysis, special characters should be avoided (but in any case, they will be replaced).
- “ConditionList”: This list has to be filled with variables of interest taken from “Model.X”.
- “Weights”: Assign the weights to “ConditionList” elements;
- “Test”: Indicate if a t-test or an F-test is intended;

4. Execute *SPM_batch_creator(1)* once again. After validation of the json file, the first level batches will be created for all specified subjects and subsequently executed. All created files are saved to a subject’s sub-folder in the output directory.

A.II. Second level analysis

5. At this point, the second level node of the json file has to be partly filled:
- “Transformations.Instructions.Name”: must be “factorial_design_specification” for the second level node;
- “MainProjectFolder”: In case that any covariates shall be included in the model, this folder must contain the file “participants.tsv”.
- “Input.Design.Type”: must be one of the factorial design options offered by SPM (OneSampleTTest, TwoSampleTTest, PairedTTest, MultipleRegression, OneWayANOVA, OneWayANOVAWithinSubject, FullFactorial, FlexibleFactorial)

- Each of these eight options require different inputs. For this reason, the template contains eight separate sections to discriminate which fields are required for the respective method. Unused sections should be deleted, but are ignored anyway (according to "Input.Design.Type").
- Details about which inputs are required for each of the options are given below (chapter A.II.1).
- "Covariates" are taken from the required file containing information of all participants. It is searched for in the "MainProjectFolder" by a regular expression. For more flexibility, it is possible to use different source files for the covariates.
- Analogous to the first level analysis, "Model.X" and "Contrasts" are left empty.

6. Execute *SPM_batch_creator(2)* for the first time. After validation of the json file and the given contents, the script will end with an intended error:

```
'NODE 2: The design matrix predictors are empty. Please check automatically
generated predictors and update the contrasts section with them.'
```

7. Complete the second level node by filling the "Contrasts" and "DummyContrasts" sections.
8. Execute *SPM_batch_creator(2)* once again. After validation of the json file, the second level batches will be created for all specified contrasts and subsequently executed. All created files are saved to subfolders of the output directory given for this node.

To keep track of the execution steps and any warnings that might occur, a logfile is stored to the "OutputDirectory". If this file is not needed, logging can be skipped by adding 0 as a second argument of the function call (i.e., *SPM_batch_creator(1,0)* or *SPM_batch_creator(2,0)*).

A.II.1 Detailed description of required second level inputs

In the second level node, the output files (i.e., contrast files) of the first level node are further processed. According to BIDS [2], any "Contrasts" defined in the previous node are available in the next one. Three options are available for definition of the input file type: "con", "beta, and "other". In order to comply with the BIDS specification, input file type <u>must</u> be "con" (default). Using this option, con_####.nii files are taken as the input images. To prevent mismatching files among subjects (e.g., due to missing contrasts if events were missing), they are searched for by their names in SPM.mat (SPM.xCon.name), which is identical to the Contrast name in the json file. Those names have to be inserted in "ContrastsToProcess" for second level processing. Additionally, any "DummyContrast" can be selected by adding its identifier taken from DummyContrasts.Contrasts.
The software offers the possibility to alternatively use the beta files of the first level analysis as input. In some cases this might be useful, when definition of "DummyContrasts" was omitted to save disk space or if simple contrasts shall be brought to the second level that were not previously defined during first level processing. (The con files of "DummyContrasts" are identical to the respective beta files.) Similar to the con files, beta files are searched for by their name (SPM.VBeta.descrip). As beta files are created for each column of the design matrix, the "ContrastsToProcess" have to be taken from Model.X of the first level node. However, the actual design does not allow to use both con and beta file types in the same model.
Furthermore, the option "other" allows to search for the input files by a regular expression. Using this option, "ContrastsToProcess" have to be taken from the Contrast names of the first level node, but the file search is done with the given regular expression instead of the contrast name. Importantly, each entry in the list of regular expressions is assigned to the respective entry in the list "ContrastsToProcess" (i.e., both lists need to have the same length and order). This option might be useful in case that SPM.mat is unavailable or if the user wants to assign custom files to the contrasts.

The options “One-sample t-test”, “Two-sample t-test”, and “Multiple Regression” allow for creation of several models/jobs: The chosen method will be applied to all contrasts given in “ContrastsToProcess”, and the results will be saved to separate folders named by the first level contrasts.

A.II.1.1 One-sample t-test
A one-sample t-test is applied to all contrasts mentioned in the list “ContrastsToProcess” including all subjects that were chosen for the first level analysis.

A.II.1.2 Two-sample t-test
A two-sample t-test is applied to all contrasts mentioned in the list “ContrastsToProcess”. The subjects of the first level analysis have to be assigned to the respective group using their “Participant_ID”.

A.II.1.3 Paired t-test
In the “Scans” array, each pair can be defined as a separate array element by filling the “Participant_ID” and “Contrast.Name” of the first image of a pair in “Subjects1” and “Contrasts1” and similarly for the second image of a pair in “Subjects2” and “Contrasts2”. To facilitate the input, several pairs can be created from a single “Scans” array element. Example:
“Subjects1”: [“sub-01”,”sub-02”,”sub-03”], “Subjects2”: [“sub-01”,”sub-02”,”sub-03”], “Contrasts1”: [“Con1”], “Contrasts2”: [“Con2”] is equivalent to:

Pair #	First image of pair	Second image of pair
1	sub-01_Con1	sub-01_Con2
2	sub-02_Con1	sub-02_Con2
3	sub-03_Con1	sub-03_Con2

More generally: “Subjects1”: [“sub-01”,”sub-02”,”sub-03”], “Subjects2”: [“sub-04”,”sub-05”,”sub-06”], “Contrasts1”: [“Con1”,”Con2”], “Contrasts2”: [“Con3”,”Con4”] is equivalent to:

Pair #	First image of pair	Second image of pair
1	sub-01_Con1	sub-04_Con3
2	sub-02_Con1	sub-05_Con3
3	sub-03_Con1	sub-06_Con3
4	sub-01_Con2	sub-04_Con4
5	sub-02_Con2	sub-05_Con4
6	sub-03_Con2	sub-06_Con4

The orders within the arrays “Subjects1” & “Subjects2” and “Contrasts1” & “Contrasts2” are crucial!

A.II.1.4 Multiple regression
This option allows the inclusion of covariates in the model, which need to be stored in the file “participants.tsv” (located in the “MainProjectFolder”).

A.II.1.5 One-way ANOVA (between subjects)
Different groups of scans are arranged in “Cells”: A single contrast from the first level analysis has to be entered in “ContrastToProcess” and all associated subjects added in the “Subject_ID” list.

A.II.1.6 One-way ANOVA – within subject
Using this method, different conditions (i.e., associated contrast images) from the same subject are compared. In the “Subjects” array, several subjects can be entered in “Subject_ID”, if the

“ContrastsToProcess” are representing the same conditions (order!). Thereby, the first Contrast is linked to Condition 1, the second to Condition 2, etc.

A.II.1.7 Full factorial
This option allows to consider more than one factor (i.e., independent input variables). These are defined in the “Factors” array by their name, the number of levels, and some other options. Within the “Cells” array, the “ContrastToProcess” along with all belonging subjects and the associated levels of the previously defined factors have to be provided. For example, a “Levels” vector of [2,1] means that the images of this cell belong to the second level of factor1 and the first level of factor2. When activating the option “GenerateContrasts”, the contrasts necessary to test for all main effects and interactions will automatically be generated (without an additional “Contrast Manager”).

A.II.1.8 Flexible factorial
This option as well allows to consider more than one factor (i.e., independent input variables). Their definition requires the “Name” and selection of some options (in “Factor” array). In the “Subjects” array, all contrasts that should be considered, are entered in “ContrastsToProcess”. The “Conditions” array/matrix contains the levels each of the “ContrastsToProcess” belong to. For example, having defined two factors and “ContrastsToProcess”: [“Contrast1”,”Contrast2”,”Contrast3”] requires a “Conditions” matrix of size 3x2 (each row dedicated to one contrast, each column to one factor). “Conditions”: [[1,1],[2,1],[1,2]] in this context means that Contrast1 belongs to the first level of Factor1 and the first level of Factor2, while Contrast2 belongs to the second level of Factor1 and the first level of Factor2, while Contrast3 belongs to the first level of Factor1 and the second level of Factor2. If the “Conditions” matrix is the same for several subjects, they can all be entered in the array “Subject_ID”. (Otherwise, a separate array element in “Subjects” is required.)
In the array “MainEffectsAndInteractions”, all main effects that should be included, are added in a single array element with “Type”: “MainEffect”, giving all “FactorNames” in the respective array. For each interaction, an additional array element is required in “MainEffectsAndInteractions” (with “Type”: “Interaction” and “FactorNames” containing the two factor names which interact).

Part B: Generation and processing of json model files using the (optional) user interface *ui_spm_batch_creator.m*

The user interface supports the user in the preparation of the json model file by asking for the relevant information directly and creating the json file automatically. The main script of the user interface is *ui_spm_batch_creator.m*. Generally, three templates are available there (“First level”, “Second level”, “First + Second level”), which ask the user for certain information needed to create the json (and spm batch job), when clicking through several input windows. In most of the windows, there is an option to “go back” to the previous window by checking the respective box and clicking “OK”. However, windows which are displayed repeatedly in a row (e.g., definition of a series of conditions), usually do not offer to go back directly (or only on its last appearance), but in this case it is possible to go back from a later window. Most of the inputs are stored temporarily, so that going back and forth does not require the user to re-enter everything. Clicking “Cancel” or pressing “Esc” aborts script execution.
The template selection in the very first dialog box defines which nodes are attached to the json file. While the option “First level” only creates this node, the option “First + Second level” additionally creates the second level node based on the data of the first. The option “Second level” appends the second level node to an existing json file that was previously generated by the option “First level” (or

manually created). This last option also allows to create several second level models based on the same first level node (saved in different json model files). Screenshots of the input windows and some explanations are summarized in SequenceUI.pdf.

B.I. Template selection: "First level"

This modality is used to create a json file for the first level analysis of the data. The user is asked to provide information about the fMRI data, the design of the experiment, the model, and some general options. Two sub-cases are possible:

(a) If the user chooses to define contrasts (in the window 'Model estimation'), the provided input is checked by analyzing the data and writing the Model.X array and HRF variables to the json file. The Model.X array contains all design matrix column names, i.e., all predictors included in the model. As contrast definition is based on the content of Model.X [2], the provided input data have to be validated and analyzed first. The user is then asked to define contrasts in the window 'Contrast definition'. The final json file (containing only a first level node) is written to the specified output directory. The json file can then be used to run the SPM_batch_creator to generate and execute SPM batch files by prompting in MATLAB (the number argument indicates the level of statistical analysis):

SPM_batch_creator(1);

(b) If the user chooses not to define contrasts, the json file is written directly after the 'Save model file' window. This option is quicker since the data is not analyzed yet. Instead, the contrast sections are empty and the user has to define contrasts manually in the json file. For this purpose, the user has to execute *SPM_batch_creator(1)*, which will result in an error message, since the predictor variables are not defined yet. However, the sections Model.X and Model.HRF.Variables are filled and those entries can then be used to define contrasts manually in the json file. Afterwards, the user can execute *SPM_batch_creator(1)* again to generate and execute SPM batch files.

B.II. Template selection: "Second level"

This modality is used to create the second level node of the json file, which will be appended to the nodes structure of an existing json file. This option requires a first level json file (created with the option 'First level analysis') as input. The user is asked to provide information about the second level analysis, such as the design of the experiment, the model, and some general options. The json file is written to the specified output directory. Two sub-cases are possible:

(a) If the user chooses to define contrasts (in the window 'Second level input'), the provided input is checked by analyzing the data and writing the X array to the json file. The user is then asked to define contrasts in the window 'Contrast definition'. The final json file (containing a first and second level node) is written to the specified output directory. The json file can be used to run the SPM batch creator to generate and execute SPM batch files by executing:

SPM_batch_creator(2);

(b) If the user chooses not to define contrasts, the json file is written directly after the 'Estimation Options' window. This option is quicker since the data is not analyzed yet. Instead, the contrast sections are empty and the user has to define contrasts manually in the json file. For this purpose, the user has to execute *SPM_batch_creator(2)*, which will result in an error message, since the predictor variables are not defined yet. However, the section Model.X is filled and this entry can then be used to define contrasts manually in the json file. The user can then execute *SPM_batch_creator(2)* again to generate and execute SPM batch files.

B.III. Template selection: "First + Second level"

This modality is used to create a json file for the first and second level analysis of fMRI data. The user is asked for the same information as in the first level analysis, but is forced to define contrasts in the window 'Contrast definition'. This is necessary since the second level analysis is based on the contrasts defined in the first level analysis. The json file is written to the specified output directory. The second level definition is done in the next step (see B.II.). In this case, it is not possible to define contrasts in the user interface of the second level analysis, since the required variable Model.X cannot be defined yet for the second level node (because the contrast files of the first level analysis are not yet available). The user has to define contrasts manually in the json file. For this purpose, the user has to execute *SPM_batch_creator(1)* to generate and execute SPM first level batch files and then *SPM_batch_creator(2)*. This results in an error (because Model.X is not defined), but this variable is at this point generated allowing the user to define the second level contrasts manually. Finally, *SPM_batch_creator(2)* has to be executed once again to generate and execute SPM second level batch files.

B.IV. Recommended use

The most convenient way to run *ui_spm_batch_creator* and *SPM_batch_creator*, i.e., manual input in the json file is not necessary, is to use this function in the following way:

i) Run *ui_spm_batch_creator* with option 'First level analysis' and activate 'Check data to define contrasts' in the window 'Model estimation' (see B.I.a).

ii) Execute *SPM_batch_creator(1)* to generate and execute SPM first level batch files.

iii) Run *ui_spm_batch_creator* with option 'Second level analysis' and activate 'Check data to define contrasts' in the window 'Second level input' (see B.II.a).

iv) Execute *SPM_batch_creator(2)* to generate and execute SPM second level batch files.

Part C: Handling of missing data

In general, if the software cannot find data in the expected location or if the json file contains incompatible fields, a warning message will appear in the command window. Usually, these warning messages should tell the user exactly, what is missing or what would have been expected. Warning messages do not necessarily mean that batch creation or batch execution will fail, since some mechanisms are included in the scripts, which can handle at least some scenarios:

- Events (like events.tsv) files are expected in each subject's raw data directory, but are also searched for in the preprocessed folders if not found in raw directory.
- If several files are found using a regular expression while expecting just one, filters will be applied in case that additional information is available (like task, session, run). If that does not help to determine a single result, the software uses the first one. Therefore, one has to act with caution, if such warning occurs.
- If pre-defined conditions/events are missing in some subjects, related contrasts cannot be created and the respective subjects will be excluded from second level analysis for this specific contrast. The resulting mismatch in enumeration is circumvented by using the contrast names instead.
- Missing movement parameters are replaced by zeros.
- A missing parametric modulator value for a single event leads to omission of that event. Missing parametric modulator values for all events leads to omission of the parametric modulator, but keeping all events.

- Similarly, a missing covariate entry in second level analysis, leads to exclusion of the affected subject.
- If covariates (or parametric modulators) are stored as categorical entries (e.g., gender as F, M, D), the categories will be converted to numbers (in order to be accepted by SPM12). However, it is recommended to use numbers already in the input files to prevent any uncertainty about the consistency of the number representations. Any conversion of categories to numbers leads to a warning message and to an info line about the conversion.
- Participants files (like participants.tsv) are expected in the main project folder, but are also searched for in subdirectories if not found in the main directory.

If warning messages occur during data validation, a final user request is stated, informing that 'Warning(s) occurred (see command window)'. The user can decide to 'abort' execution (and fix the inputs) or to 'ignore' the warnings (and try to create and execute the batch job regardless).
As default, all warning messages and other outputs in the command window are stored to a logfile in the "OutputDirectory". If this file is not needed, logging can be skipped by adding 0 as a second argument of the function call (i.e., *SPM_batch_creator(1,0)* or *SPM_batch_creator(2,0)*).

Part D: Default values

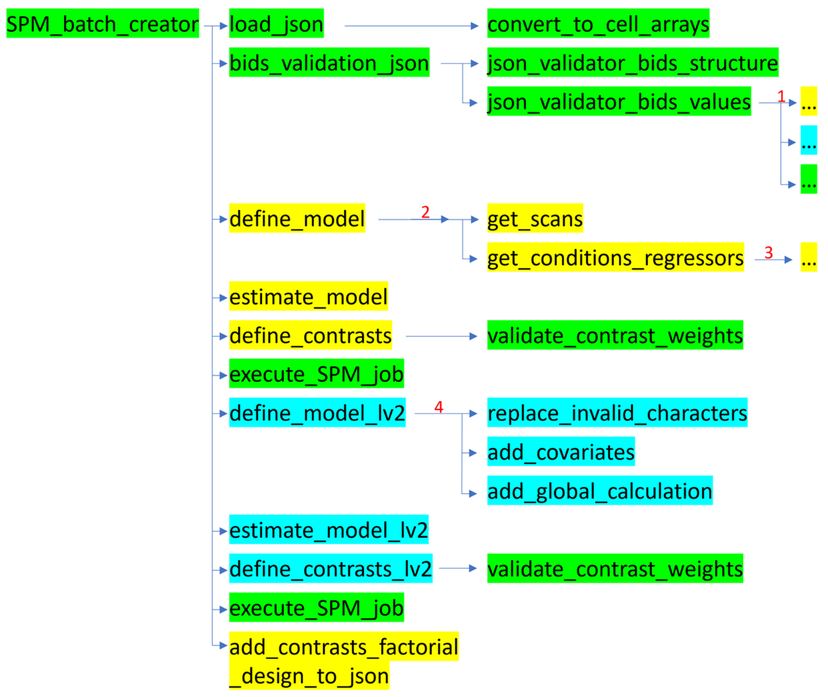
Some default values, which are collected in the file *get_default_values.m*, can be adjusted by the user:

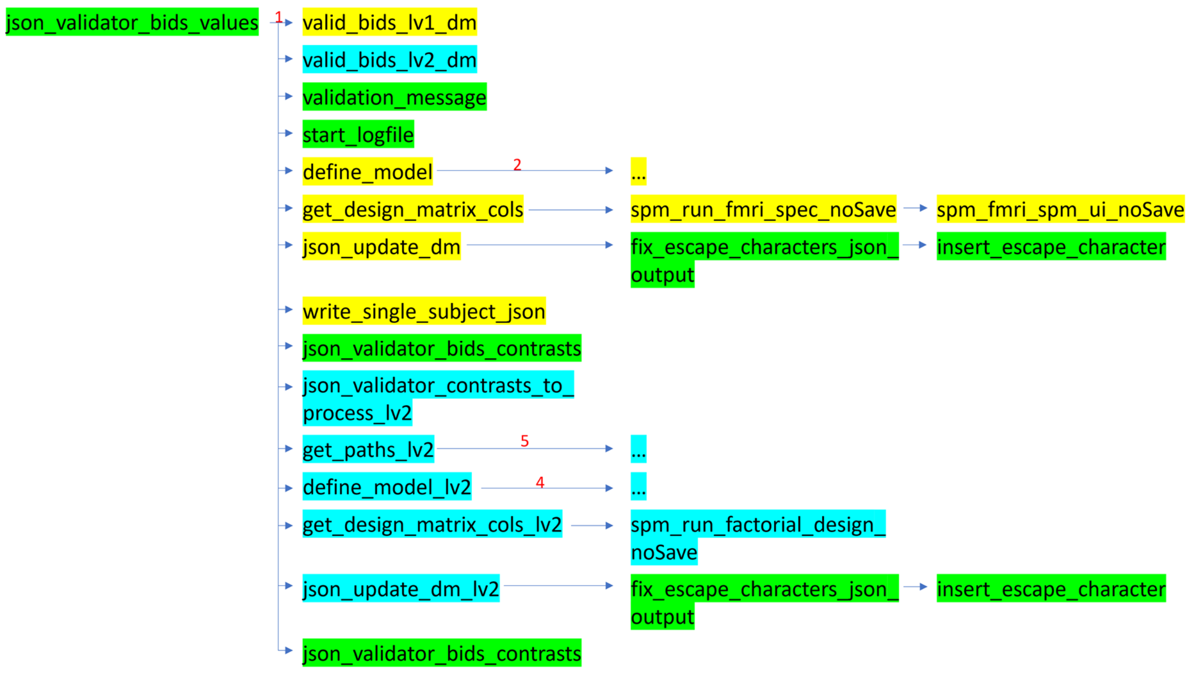
- missing_data_indicator: cell array containing strings that should be interpreted as missing data in events.tsv and participants.tsv
- raw_data_dir, preproc_data_dir, first_level_dir, second_level_dir: default folders for non-BIDS datasets, where the respective data are stored (see Fig. 3)
- In the part referring to the user interface (*ui_spm_batch_creator.m*), the following variables may be subject to change:
    - default names for first and second level nodes
    - different separators for ui text inputs
    - masking thresholds
    - overall grand mean scaling value

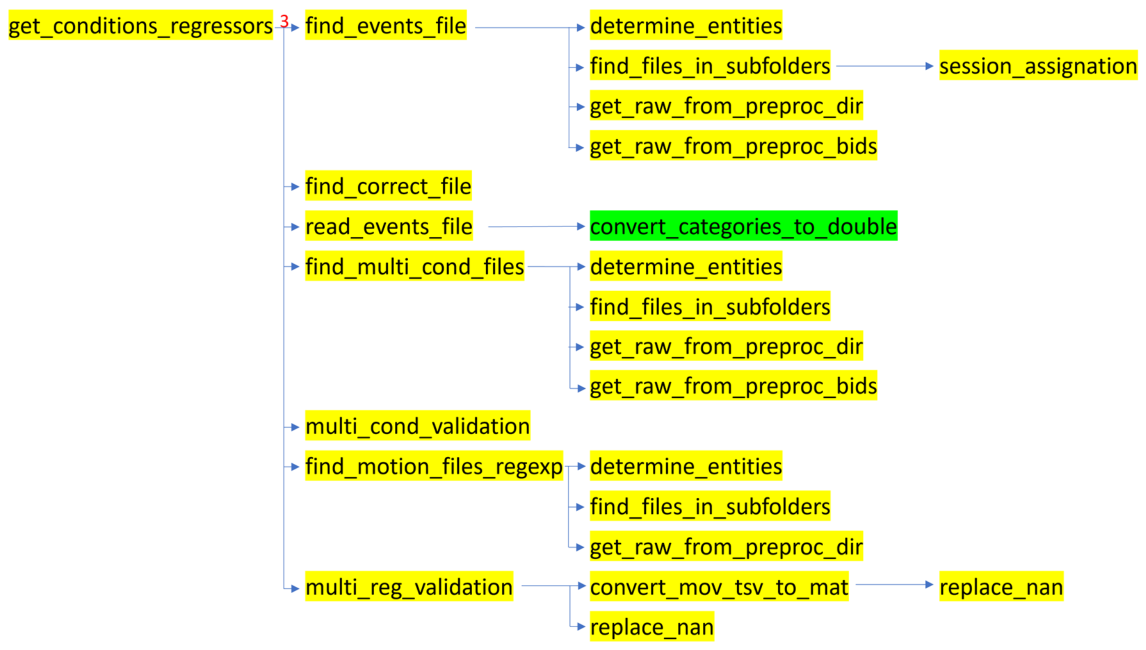
APPENDIX

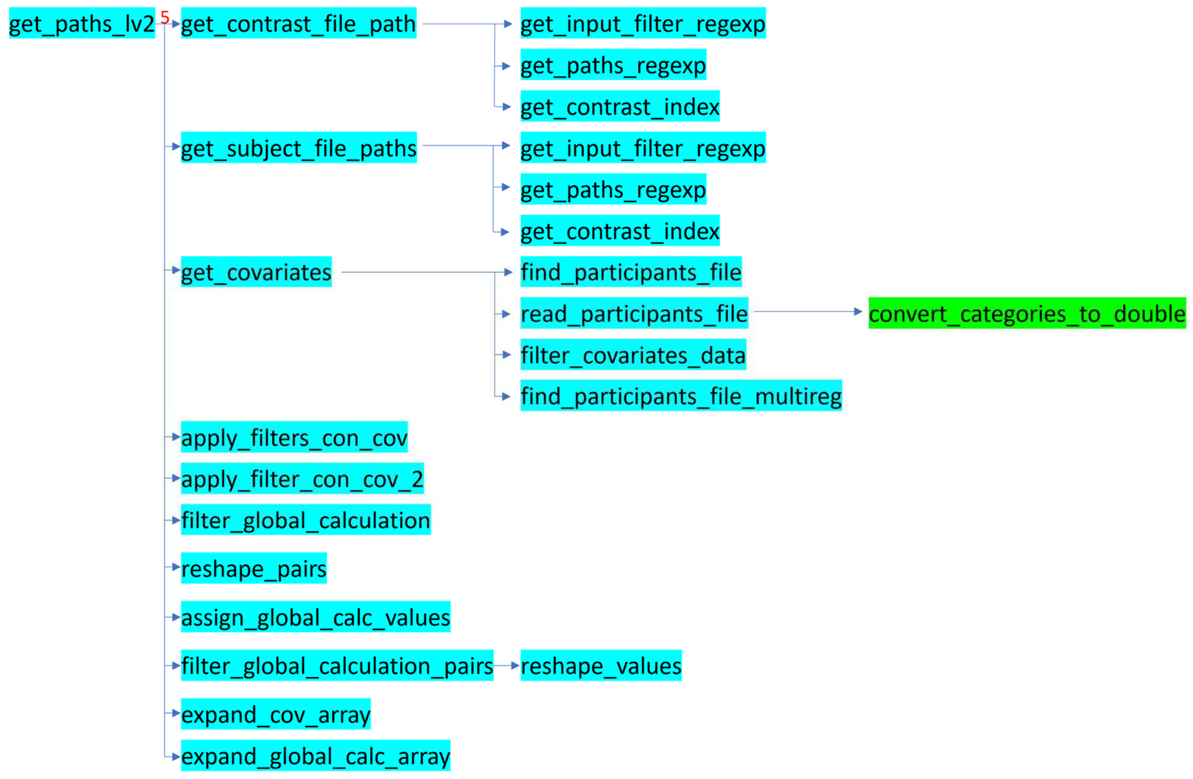
A. Workflow within the software

The following images show the workflow within the scripts. Number references in the images represent links between the respective images. Script names highlighted with yellow indicate first level related scripts, blue highlighting refers to second level scripts, and green highlighting indicates reference to both levels of analysis.




json_validator_bids_values
1
valid_bids_lv1_dm
valid_bids_lv2_dm
validation_message
start_logfile
define_model
2
...
get_design_matrix_cols
spm_run_fmri_spec_noSave
spm_fmri_spm_ui_noSave
json_update_dm
fix_escape_characters_json_output
insert_escape_character
write_single_subject_json
json_validator_bids_contrasts
json_validator_contrasts_to_process_lv2
get_paths_lv2
5
...
define_model_lv2
4
...
get_design_matrix_cols_lv2
spm_run_factorial_design_noSave
json_update_dm_lv2
fix_escape_characters_json_output
insert_escape_character
json_validator_bids_contrasts


get_conditions_regressors
3
find_events_file
determine_entities
find_files_in_subfolders
session_assignation
get_raw_from_preproc_dir
get_raw_from_preproc_bids
find_correct_file
read_events_file
convert_categories_to_double
find_multi_cond_files
determine_entities
find_files_in_subfolders
get_raw_from_preproc_dir
get_raw_from_preproc_bids
multi_cond_validation
find_motion_files_regexp
determine_entities
find_files_in_subfolders
get_raw_from_preproc_dir
multi_reg_validation
convert_mov_tsv_to_mat
replace_nan
replace_nan


get_paths_lv2
5
get_contrast_file_path
get_input_filter_regexp
get_paths_regexp
get_contrast_index
get_subject_file_paths
get_input_filter_regexp
get_paths_regexp
get_contrast_index
get_covariates
find_participants_file
read_participants_file
convert_categories_to_double
filter_covariates_data
find_participants_file_multireg
apply_filters_con_cov
apply_filter_con_cov_2
filter_global_calculation
reshape_pairs
assign_global_calc_values
filter_global_calculation_pairs
reshape_values
expand_cov_array
expand_global_calc_array

References:

1. A. Oliver-Taylor et al., The Brain Imaging Data Structure (BIDS) Specification (v1.10.0) (2024), https://zenodo.org/records/13754678
2. A. de la Vega et al. BIDS Extension Proposal 2 (BEP002): The BIDS Stats Models Specification (version 1.0.0-rc1) (2022), https://docs.google.com/document/d/1bq5eNDHTb6Nkx3WUiOBgKvLNnaa5OMcGtD0AZ9yms2M/edit?tab=t.0#heading=h.mqkmyp254xh6