Chapter 7

Actionable Event Annotation and Analysis in fMRI: A Practical Guide to Event Handling

Monique J. M. Denissen, Fabio Richlan, Jürgen Birklbauer, Mateusz Pawlik, Anna N. Ravenschlag, Nicole A. Himmelstoß, Florian Hutzler, and Kay Robbins

Abstract

Many common analysis methods for task-based functional MRI rely on detailed information about experiment design and events. Event recording and representation during cognitive experiments deserves more attention, as it forms an essential link between neuroimaging data and the cognition we wish to understand. The use of standardized data structures enables tools to directly use event-based metadata for preprocessing and analysis, allowing for more efficient processing and more standardized results. However, the complex paradigms utilized by cognitive neuroscience often have different requirements for event representation. The process of generating event files from experimental logs and to iteratively restructuring these event files is a time-intensive process. Careful planning and effective tools can reduce the burden on the researcher and create better documented and more shareable datasets. This chapter discusses event representation within the BIDS (Brain Imaging Data Structure) framework. We discuss some of the common pitfalls in event representation and introduce tools to easily transform event files to meet specific analysis requirements. We demonstrate these tools and the corresponding analysis by comparing two BIDS datasets in which participants performed a stop-signal task. We work through the required event restructuring, and use *Fitlins* to calculate several comparable contrasts across the two datasets.

Key words fMRI, BIDS, Events, Event annotation, HED

1 Introduction

In task-based fMRI, the most common type of analysis models BOLD signal response under different conditions to identify brain areas associated with those conditions. In such analyses, conditions are usually defined by aspects of experimental events that are relevant to human experience, cognition and behavior; the neural correlates of which we wish to understand. Thus, experimental events provide an essential link between the neural activity captured in the BOLD signal and the human experience and/or behavior we are trying to understand. Experimental events are direct drivers of

Robert Whelan and Hervé Lemaître (eds.), *Methods for Analyzing Large Neuroimaging Datasets*, Neuromethods, vol. 218, https://doi.org/10.1007/978-1-0716-4260-3_7,

this experience. Because so much of the primary analysis for fMRI is directly based on comparison of conditions—encoded in the dataset events—the accuracy, completeness, and structure of the reported events can dramatically affect the usability of the data and the correctness of the final results. Unfortunately, there is no generally accepted standard for how events should be encoded, reported, and documented.

When an experiment is conducted, information about the sensory presentations, participant responses, and other control information is usually orchestrated by stimulus presentation software such as Presentation® (Neurobehavioral Systems, Inc., Berkeley, CA, www.neurobs.com), E-Prime [1], or PsychoPy [2]. Logged information is often user-controlled, and different software packages generate log files containing different structures and data types. The experimental logs, along with the fMRI recordings and auxiliary data such additional anatomical MRI recordings comprise the raw data of an experiment, which then must be converted into a standardized format for uploading to repositories or for input to analysis tools.

The Brain Imaging Data Structure (BIDS) [*see* Chapter 4] provides a standard for organizing and storing neuroimaging data along with event information, the focus of this chapter [3]. Event files in BIDS are tab-separated value (*.tsv*) files. Each row in an event file represents an event, and each column represents a different aspect of the event. BIDS only requires *onset* and *duration* columns in event files. The *onset* gives the time of the event marker in seconds relative to the start of the file's associated data recording, and the *duration* gives the duration of the event in seconds. Researchers may describe other properties of events by including additional columns in the event files. These additional columns and the values contained in them may be described in accompanying JSON files [*see* Glossary] (sidecars), but these sidecars are not required.

BIDS also supports Hierarchical Event Descriptors (HED) [*see* Chapter 6] for creating machine-actionable annotation [4, 5]. HED is a vocabulary designed to describe experiment events in a structured human-readable and machine-actionable way. This means events annotated with HED can be searched across datasets and understood outside of the context of any specific study. A HED annotation is a string of HED terms that is associated with an event in a BIDS event file. Terms are organized hierarchically to allow for search on broad categories as well as specific terms. However, the addition of HED is optional in BIDS, and so a BIDS-compliant dataset does not necessarily contain any documentation of events.

BIDS is now a well-established standard for storing neuroimaging data. Neuroimaging repositories such as OpenNeuro [6] [*see* Resources] expect datasets to be in BIDS. *BIDSApps* provide data management and analysis tools for BIDS datasets, leading to more

standardized data processing and more reproducible neuroscience [7]. These apps are powerful because they can utilize standardized APIs (application program interfaces) to automatically access metadata such as scanning parameters or event data as necessary. By having a single software tool format, analyses stay flexible while also becoming more accessible, open, and reproducible. For the use of these tools, however, the *quality* of the data stored in BIDS is essential. This book chapter focuses on how to manage events and prepare event files for analysis in the context of requirements for downstream automated analysis.

To illustrate the process of generating extensive and analysis-ready event files, we demonstrate a simple analysis of two datasets: AOMIC-PIOP2 available on OpenNeuro (ds002790) [*see* Chapter 2] and SOCCER [https://osf.io/93km8/]. Both of these employed a stop signal task. In order to process both datasets and make them comparable, we carefully consider the event files, their structure, and the potential pitfalls that occur during setup.

Figure 1 illustrates a typical lifecycle of data from a neuroimaging experiment. To start, researchers use a combination of the data recordings and the experimental log files to convert their datasets to BIDS. The log files from the presentation software packages are

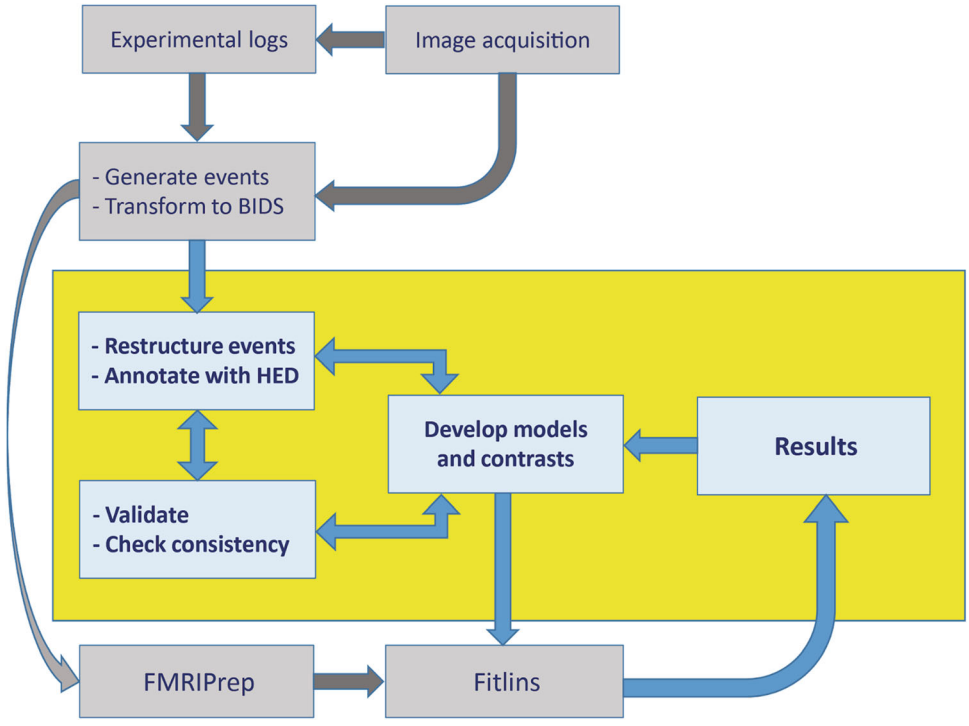


Fig. 1 The lifecycle of data from a neuroimaging experiment. The yellow box is the focus of this chapter. *FMRIPrep* and *FitLins*, which are the widely used open-source preprocessing and analysis tools used in this chapter, are placeholders for preprocessing and analysis, respectively

used to generate the experimental event files. When obtaining a BIDS dataset from an external source this process will have already been completed. The two example datasets demonstrate these respective use cases.

The AOMIC-PIOP2 dataset is part of a large set of data collected at the University of Amsterdam and made available on OpenNeuro. For this dataset, the analysis must rely only on the event data available on OpenNeuro plus any published descriptions of the dataset available in related publications [8]. This is the typical situation for researchers using open repositories to obtain data for analysis.

The second dataset, SOCCER, contains data collected at the University of Salzburg and provides a more typical use case for researchers working with their own data. Here we start directly from the experimental log files, allowing revision and clarification of the event encodings taken from experiment logs as would be the situation for the original experimenter who is working on initial publications and release of the data. After generating a fully BIDS compliant dataset that has been adequately preprocessed, we can start the linear modeling using *FitLins*. However, as we will show, effective linear modeling often requires event files to be restructured while executing the desired models and checking the results.

Preprocessing is by far the most computationally intensive aspect of the life-cycle. For preprocessing, we have chosen to use *FMRIPrep* [9] [*see* Chapter 8], a standard package for fMRI preprocessing. *FMRIPrep,* which is available as a *BIDSApp* [7] [*see* Glossary], relies only on the BIDS imaging data and does not use the event files. Thus, it can be run once for the dataset independently of the event processing and downstream analysis. For linear modeling we have chosen *FitLins* [10], which is also available as a *BIDSApp* and performs basic multilevel linear modeling. We have chosen *FitLins* because of its suitability for large-scale deployment and analysis and because of its close integration with the newly emerging BIDS Stats Models [11]. The BIDS Stats Model allows specification and validation of the modeling process using a JSON file specification. We believe this type of model specification is important for the documentation and reproducibility of analyses. Note that although we have chosen to use specific *BIDSApps*, instructions on handling of event files are not specific to the use of these analysis tools, or even to the use of *BIDSApps* in general.

As indicated in Fig. 1, the linear modeling process (*FitLins*) is closely integrated with event processing in a feedback loop. The fMRI modeling often relies on using discrete values in an event file column as contrasts [*see* Glossary] in a linear model. For a modeler wanting to ask the exact question envisioned by the person who created the dataset, using these discrete values might be possible, but in most cases it is unlikely to be sufficient. Hence, modelers will need to rework their event files, either through their own coding or

through the emerging pybids-transforms [*see* Resources] mechanism [12], in order to perform significant analysis.

Thus, a feedback cycle emerges in which a model is developed, results are reviewed, and potentially additional reorganization of event files is done to enable additional analysis. While preprocessing of the imaging data is computationally intensive, it can be automated. In contrast, the event restructuring and model development is researcher-time intensive and challenging; however, it can be made more effective by better community guidelines and more tools for user-friendly changes and updates to event files. Here we present some guidelines on what to consider when building event files. We also discuss some open-source tools that we have developed to help to fix issues when they are encountered.

It is possible to follow along with the demo data. In the following section we will first go over the requirements for working along with the tutorial. We will describe the datasets used in the tutorial and highlight the differences that we will later compare in the example analysis. Next, we will discuss the layout of the initial event files for both datasets and issues we encountered as a consequence of this layout, along with general recommendations for the event file structure. We then briefly discuss the BIDS Stats Model and explain how it bridges between BIDS data and linear analysis of this data. Based on the issues we encountered in the event files, we present a series of tools that can be used to restructure event files for analysis. We illustrate the use of these tools, the BIDS Stats Model, and *FitLins* on our example data. The demo data and supporting material for this chapter are available at [https://osf.io/93km8/]. This chapter emphasizes event processing from the perspective of fMRI analysis, but the general concepts also apply to other imaging modalities such as EEG and MEG.

2 Methods

2.1 Starting Point for Data

The analysis requires a dataset in BIDS format as would be downloaded from OpenNeuro [https://openneuro.org] [*see* Chapter 2], or transformed directly from collected data. At least two subjects are required, since multilevel analysis is discussed. Clearly, using a larger number of subjects is essential to obtain adequate statistical power for reproducible group level results, but statistical power is not the focus of this chapter. One should also be aware—as we shall demonstrate—that getting processing to work with a small number of subjects does not guarantee that this processing will work with many subjects due to variances in the runs and missing or bad data.

We assume that the data is preprocessed using *fMRIPrep*, which produces files in a BIDS-derivative compliant format that can be automatically ingested by *FitLins* without intervention (see the supplementary material for additional information on

usage [https://osf.io/93km8/]). Issues such as inconsistency or missing values in the events are part of the event restructuring process addressed in this chapter.

2.2 Data Storage and Computing

fMRIPrep and *FitLins* are—in theory—platform-independent. Because our demos involve small datasets, the computations can be done on a desktop computer (although *FMRIprep* may take several hours per subject depending on the processing power of the desktop computer being used). *FitLins* for a small number of subjects usually takes less than an hour on a moderately powered desktop computer.

The three-subject AOMIC-PIOP2 dataset that we have provided as a demo [https://osf.io/93km8/] has raw data of approximately 220 MB and processed data derived from *FMRIPrep* of approximately 1.12 GB. The processed results from *FitLins* for this dataset are about 50 MB for each model. The computing time and storage required for event processing and restructuring is negligible.

2.3 Software and Coding

Some coding experience is necessary. Running of *fMRIprep* can be done by typing a single command, which does not require coding skills but does require an understanding of command-line arguments and their meaning. *FMRIprep* is available as a Docker image, but it can also be installed as a python package to run without Docker.

In theory, *FitLins* only requires knowledge of command-line arguments and their meaning as well as access to a Docker-enabled system. *FitLins* also requires a model file in JSON format, which can be created using an ordinary text editor. The construction of the JSON file requires knowledge of how the models are encoded using the BIDS Stats Model. The interaction between these model files and the event files is the focus of this book chapter.

The tools that we have developed for event structuring and re-coding are written in Python and assume at least Python version 3.7. These tools are available for support of event processing with minimal programming. These tools are being integrated in the Hierarchical Event Descriptors (HED) tool suite [https://www.hed-resources.org/] for support of event handling and annotation [*see* Chapter 6]. These tools will soon be available via a command-line interface in a Docker container and will also be available as web services and through online tools with no coding or software installation [https://hedtools.ucsd.edu/hed].

2.4 Framing the Problem

This chapter uses the problem of comparing results from two datasets (AOMIC-PIOP2 and SOCCER) on a particular task (stop-signal) as a motivation and a focus for discussing event restructuring with an end goal of comparing results across these experiments.

2.4.1 The Task

The stop-signal task is a well-studied test for understanding response initiation and inhibition [13, 14]. In a series of trials [*see* Glossary], participants are asked to respond to frequent "Go" stimuli but to cancel their response when a particular stop signal is presented at an interval (the "stop signal delay") after the Go stimulus is presented. The selection of the Go stimulus is the primary discrimination task, with the stop-task as a secondary detection task. Although the task structure and instructions adhere to a strict format, there can be considerable variation possible in stimuli used and the selection criteria for the primary task as illustrated by the two datasets (AOMIC-PIOP2 and SOCCER) discussed in this chapter.

AOMIC-PIOP2

In AOMIC-PIOP2, participants performed four different tasks during each of four functional runs in a single session. Participants also completed several demographic and psychometric questionnaires, either before or after scanning, on the same day. Scanning lasted about 60 min, and the entire stop-signal run lasted around 7 min.

The primary discrimination task was a gender decision task using face images. The stop signal was an auditory stimulus, presented with a delay starting from 250 ms but shortened stepwise if the average number of failed stop trials was higher than 50% and lengthened if the average number of failed trials was lower than 50%. Participants performed a total of 100 trials, as well as approximately 10 additional null trials in which no stimuli were presented for an average trial duration of 4 s.

SOCCER

In SOCCER, the stop-signal task was the only task participants performed in the MRI scanner. Scanning lasted around 45 min during which the stop-signal task was performed during a single 12-min run. This run was organized into six experimental blocks of 64 trials each, with 16 s of rest between each block. At the end of the six blocks, participants completed a survey in which they described the strategy they used during the task by indicating agreement on a 1–100 scale to several questions. The same controller buttons were used to record responses during the survey portion as during the task.

The primary selection task consisted of selecting between left and right pointing arrows, and pressing a left or right button accordingly. The stop signal was a color change in the presented arrow. The delay was updated constantly, based on a thresholding procedure designed to keep stop-signal performance around 50%. Note that although this is similar to the aim in the AOMIC-PIOP2, the exact rules to get to a 50% average were different. Also in contrast to the AOMIC-PIOP2 study, which had a broader focus, participants in SOCCER were familiarized with the task extensively by practicing at home as well as one additional time inside the

scanner. The practice block in the scanner had trial-by-trial feedback.

2.4.2 Dataset Acquisition

The two datasets represent the two most common use cases in analyses of neuroimaging—downloaded versus locally acquired data. The AOMIC-PIOP2 dataset is a public dataset, and the information about the events is restricted to that available from the dataset itself and the referenced papers. There is limited opportunity to ask for clarification or correction. SOCCER is an ongoing experiment at the time of this writing, so the experimental logs and control scripts are available, as is access to the experimenter.

AOMIC-PIOP2

AOMIC-PIOP2 is part of a large open collection from the University of Amsterdam which has been deposited on OpenNeuro (ds002790) [*see* Chapter 2]. The collection includes diffusion-weighted images and fMRI runs from four different tasks, but this chapter only considers fMRI data from the stop-signal task. Extensive details on scanning parameters, demographic and other participant characteristics are provided in Snoek et al. (2021) [8], and some information at the participant level has been distributed with the dataset. All participants of the AOMIC-PIOP2 were students, aged between 18 and 26 years old.

SOCCER

SOCCER is part of an ongoing experiment from the University of Salzburg. In this experiment male adolescent (16–17 years old) amateur soccer players are invited to participate in a study on the link between performance, development, and low-level response and inhibition processes. Participants are scanned during a single session. Functional and structural neuroimaging data were collected with a Siemens Magnetom Trio 3 T Scanner (Siemens AG, Erlangen, Germany) using a 64-channel head-coil. Functional images consisted of a T2*-weighted gradient echo EPI sequence (TR 1050 ms, TE 32 ms, matrix 80 × 80, FOV 192 mm, flip angle 45°). Within the TR 56 slices with a slice thickness of 2.4 mm were acquired. In addition to the functional images, a gradient echo field map (TR 623 ms, TE 1 = 4.92 ms, TE 2 = 7.38 ms, flip angle 60°) and a high resolution (0.8 × 0.8 × 0.8 mm) structural scan with a T1-weighted MPRAGE sequence were acquired from each participant. Additional structural files were collected but have not been shared for the purposes of this demo. Participants spent around 60 min in the MRI scanner for the entire session. Of this time, the functional run containing the stop-signal task lasted around 12 min.

2.4.3 A Side-by-Side Comparison

Although both datasets use the stop-signal task paradigm, the experiments differ significantly. Table 1 shows a side-by-side comparison of various implementation aspects of the two experiments. The similarities and dissimilarities between the tasks determine the

Table 1
An overview comparison of the two datasets used for this chapter

Aspect	AOMIC-PIOP2	SOCCER
Participant pool	University students	Amateur soccer players
Number of participants **(Demo/Total)**	**Demo:** 3 **Total:** 226	5
Participant age **(Demo/Total)**	**Demo:** 20–24, mean: 22.67 **Total:** 18–25, mean: 21.96	16–17, mean 16.4
Participant gender **(Demo/Total)**	**Demo:** 66% female, 33% male **Total:** 57% female, 42% male	100% male
Choice images	Female (right) and male (left) faces	Left and right white arrows
Choice image duration	0.5 s	Up to 1.25 s depending on button press
Discrimination task	Gender	Direction
No go indicator	Auditory tone 450 Hz for 0.5 s	Color change (white to red)
Initial No-go delay	0.25 s	0.2 s
Go response time average	1.0243 s	0.371 s
Button press configuration	Index fingers of left and right hands	Index and ring finger of right hand
Trials per run	100 trials	384 in 6 blocks of 64 trials each
Approx trial time	2.5 s	0.57 s
Total trials	15,119 go trials 4370 successful stop and 3111 unsuccessful stop trials	1452 go trials 233 successful stops and 232 unsuccessful stop trials
% of stop trials	33%	25%
% successful versus unsuccessful stop	~58% successful stops with large variation Three subjects had no successful stops, and one subject had no unsuccessful stops. In all, 61 subjects had a low number of successful or unsuccessful stops	~50% successful stops Evenly distributed across participants
Spacing	Has 10% null trials	Rest blocks

questions we can ask about either dataset or the types of comparisons we can make to contrast the two datasets.

As Table 1 states, AOMIC-PIOP2 is a large published study with a mix of participant genders, while SOCCER is an ongoing study of all male participants tightly grouped by age. SOCCER will ultimately collect data for 55 participants.

The task implementation also varied in several respects. While the AOMIC-PIOP2 study used face discrimination as a primary decision task, SOCCER uses a simpler direction discrimination task, which is expected to exert a lower cognitive load. Face discrimination has significant social correlates, and we expect to see activations related to face processing. In contrast, the identification of left versus right arrows is unlikely to be related to specific brain areas. The stop signal modalities also differed for the two datasets: AOMIC-PIOP2 used an auditory signal, while SOCCER used a visual color cue.

Reaction times vary strongly between datasets. This can be explained by differences in other aspects of the implementation: the complexity of the discrimination task, the relative athletic capabilities of the participant pool, and the amount of practice on the stop-signal task (see also below). The response hand usage was also different in the two experiments, with the AOMIC-PIOP2 participants using index fingers on right and left hands for right and left responses, possibly inducing strong hemispheric differentials. On the other hand, participants in the SOCCER experiment use two fingers on the right hand so little hemispheric differential is expected. The distribution of trial types within the files also differed, with AOMIC-PIOP2 having several participants with an imbalance between stop and go trials, while SOCCER aggressively maintained a balance between stop and go trials in each run.

2.4.4 The Impact of Event Encoding

Within a BIDS dataset, the event files fulfill multiple functions including documenting the experiment and providing direct input to the analysis. Different analyses can put their own requirements on the organization of the event file and careful consideration of the organization of the event files is necessary. *FitLins*, for example, extracts the onsets and durations for modeling BOLD responses directly from the event files.

Foundational to building models for functional MRI data analysis is the convolution of event-based boxcar functions with the hemodynamic response function (HRF) [*see* Glossary]. The relationship between neural activity and brain hemodynamics is a well-studied topic [15]. *FitLins* uses the canonical HRF based on the one used by SPM [*see* Resources] as a standard, although there are other options available [16]. For short events in event-related experimental designs [*see* Glossary] the HRF is convolved with an impulse function. For longer events that are analyzed in a block

design [*see* Glossary] the HRF is convolved with a boxcar function. For this, as well as for analysis of individual HRFs, modeling event durations precisely is important.

For the purpose of creating optimal event files, we focus on the accurate and precise reflection of event onsets and durations. In the following sections, differences in how onsets and durations are represented in event files across datasets are emphasized, and we discuss different options for representation.

While the events in the AOMIC-PIOP2 were already encoded in a BIDS event file, the starting point for the SOCCER events is a log file generated by NeuroObs presentation software. The first step here is to read out this log file and transform it into BIDS event format. Creating a minimally valid BIDS file from a log file is easy, but in order to prepare for analysis, additional details and context information must be added.

With well curated, finalized event files, there can still be many hurdles for a researcher who wants to reanalyze the data, necessitating additional time be spent on restructuring the event file and recategorizing the events.

2.4.5 Trial-Level Versus Event-Level Encoding

Trial-level encoding represents a trial by a single line in the event file with the *onset* column value marking the time of a particular anchor event (often the primary stimulus presentation) within the trial. Other trial events are either omitted or represented in other columns as offsets (or a combination of offsets) from the event onset time. Correctly analyzing the timing of any event except the anchor event can be quite complicated and require a careful reading of the documentation and accompanying literature. Automated processing can easily fail or incorrectly interpret these offsets. The practice of trial-level encoding also encourages the skipping of the extra events that occur within the trial, which can limit the scope of the data use.

The alternative is to split the trial into multiple events. While this event-level encoding results in additional rows in the event files, it allows more flexibility in downstream analysis for contrasts anchored on different internal events in the trial. Further, durations of the events within a trial can be correctly distinguished for convolution with the hemodynamic response function (HRF), an essential step in building a model for fMRI data. However, context that relates to an entire trial can be lost.

In a hybrid approach, each individual event is represented as a row as with event-level encoding. Additional events are added before the first event in each trial with a duration representing the extent of the entire trial. This trial event can be associated with information that should be modeled on a trial level. Additional events representing the start of experimental blocks may also be included.

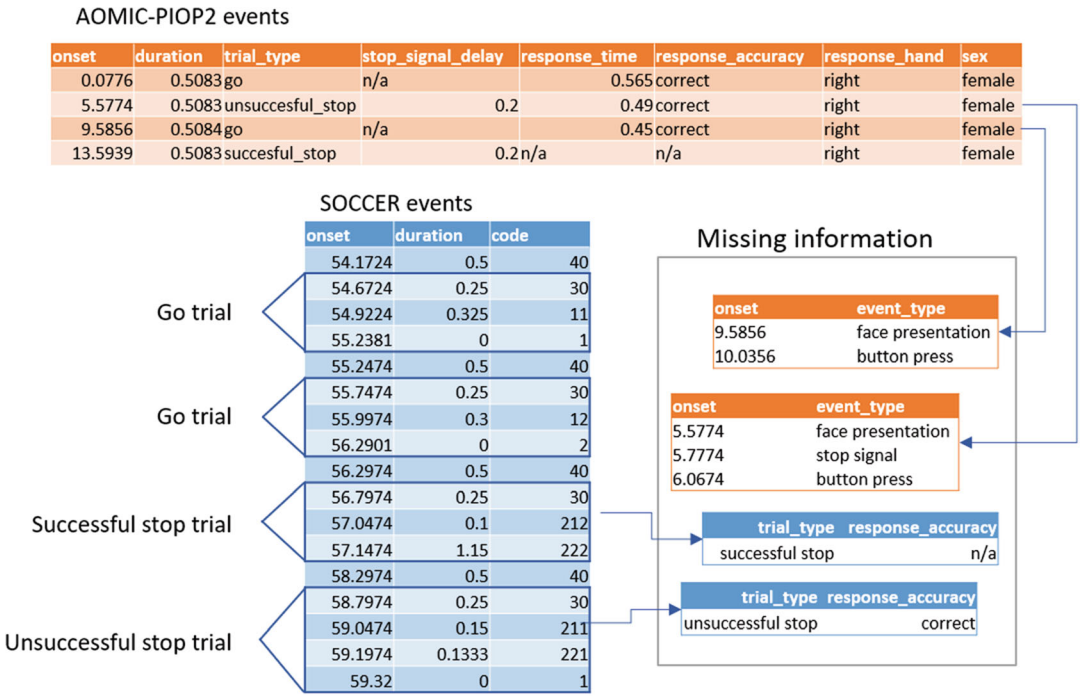


Fig. 2 Comparison of AOMIC-PIOP2 and SOCCER event file structure. Event rows in AOMIC-PIOP2 (in orange) represent entire trials with additional columns reflecting offsets of various trial events relative to trial onset. As a result, the event file onset information for stop signals and responses is implicit, as reflected in the missing information box (right box). Event rows in SOCCER (in blue) represent individual events, but information on context of the overall trial is implicit

The two datasets considered in this chapter initially used different encoding strategies, and both had to be restructured in order to perform comparative analyses. Figure 2 compares an excerpt from an event file for each dataset. The two event files represent the same number of trials, but are shaped differently because of their different encoding. The trial-level encoding used in the AOMIC-PIOP2 dataset contains information about the events in additional columns, but onsets anchored to neural data recording are missing for the stop signal and the response. In the SOCCER data, individual events are represented with onsets anchored to the neural data recording, but context information about trials is missing. In either case, restructuring the events so all information is adequately captured in the event files normally requires a programmer to manually code transformations of the event files. Later in this chapter we introduce tools that allow researchers to list the transformations needed in a JSON text file and run remodeling tools to automatically perform the transformations without additional programming.

AOMIC-PIOP2

As shown in Fig. 2, AOMIC-PIOP2 uses trial-level encoding, representing an entire trial by a single row with eight event file columns: *onset*, *duration*, *trial_type*, *stop_signal_delay*, *response_time*, *response_accuracy*, *response_hand*, and *sex*. The *onset* column corresponds to the presentation of the face for 0.5083 s and also marks the start of the trial. Other key events in a trial: the stop signal presentation and the participant response, are encoded implicitly using the *stop_signal_delay* and *response_time* as times offset from the row's *onset* column value. If there was no stop signal or no participant response in a trial, these columns are filled with *n/a*, following BIDS convention.

Trial-level encodings present two difficulties for downstream analysis. The first is that unusual events such as extra button presses cannot be represented although they elicit motor responses. The second difficulty is that these relative onsets may need to be unfolded into event markers before certain analyses (e.g., analysis linked to participant response onsets) because this information is hidden in additional columns. In some datasets, multiple offsets must be combined to get the correct time of a relative event. Without a very careful reading of the available documentation for each experiment, it may not be possible to correctly compute the position of these markers.

For example, the documentation for AOMIC-PIOP2 states that the *stop_signal_delay* and the *response_time* are both given in seconds relative to the go image presentation onset. However, the original experiment on which AOMIC-PIOP2 is based (Fig. 1a [17]) shows a *stop_signal_onset* (also of 250 ms) relative to the end of the face image presentation rather than the start of the presentation. Figure 1 of the Jahfari paper further indicates that participants were not allowed to respond until after the go image disappeared (500 ms). If the *stop_signal_onset* were actually at 750 ms rather than at 250 ms relative to the start of the trial, then the large differential in participant response times between the AOMIC-PIOP2 and SOCCER would be essentially eliminated, resulting in little potential effects of participant skill and discrimination task complexity on response time. Another possibility is that stop signals occurred 250 ms after the face image onset, but that participants were instructed not to respond until after the face image offset, since almost all response times were greater than 500 ms. While the most likely explanation for the difference in response times is the complexity of the discrimination task, the other possibilities cannot be completely eliminated since the original experimental logs are not available.

SOCCER

The SOCCER events are read out directly from the log files and distinguished by basic codes as illustrated in Table 2. Common events, such as the presentation of left and right arrows, the fixation

Table 2
Overview of common event codes used in the original SOCCER log

Code	Event description	Context string for the log
40	Blank screen	Experiment block
30	Fixation dot	
11	Left go arrow	Go trial
12	Right go arrow	Go trial
211	Left go arrow	Stop trial
221	Left stop arrow	Stop trial
212	Right go arrow	Stop trial
222	Right stop arrow	Stop trial
1	Left button	
2	Right button	
70	Block feedback	
80	Blank screen	Rest block

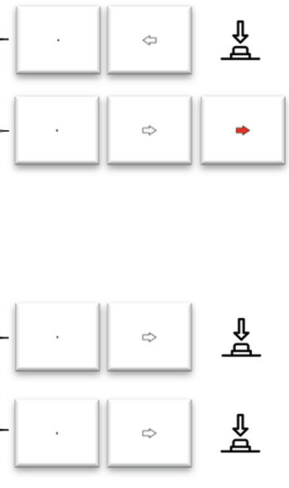
dot or a blank screen, are associated with unique numerical codes. Other codes, particularly those associated with participant responses, have different meanings depending on the phase of the experiment (i.e., the experimental context) during which the corresponding event occurred. Unambiguous coding, so important for downstream analysis, can be achieved either by reassigning ambiguous codes or by providing context marker events in the data.

In general, the log files generated by experiment control software contain markers for individual events, not for entire trials. If trial-level event encoding (as in AOMIC-PIOP2) is desired, code must be written to identify sequences of event markers in the experimental log and generate the appropriate trial events. Figure 3 shows some examples of the sequences that must be identified and collapsed.

For example, the first boxed sequence of events on the left in Fig. 3 contains the event marker sequence 30, 11, 1 starting at *onset* 340.9269 s. This sequence contains a fixation dot, followed by a left go arrow, followed by a left button press and indicates a *go* trial. The corresponding box on the right of Fig. 3 shows the insertion of an event marker representing the start of this *go* trial. The *onset* of this structure marker is the same as the *onset* of the earliest event marker in the sequence. The *duration* is the difference between its onset and the end of the last event marker in the sequence.

Once these trial event markers are inserted and the trial type is identified, it is much easier to transform to trial-level coding or to

onset	duration	code
340.4269	0.50	40
340.9269	0.25	30
341.1769	0.33	11
341.4953	0.00	1
341.5019	0.50	40
342.0019	0.25	30
342.2519	0.15	212
342.4019	1.10	222
343.5019	0.02	70
360.7230	0.00	3
360.7272	2.01	80
...	...	...
374.7858	2.01	80
378.3026	0.50	40
378.8026	0.25	30
379.0526	0.36	12
379.4028	0.00	2
379.4109	0.50	40
379.9109	0.25	30
380.1609	0.31	12
380.4648	0.00	2
380.4693	0.50	40



onset	duration	code	trial_type
340.4269	0.50	40	
340.9269	0.58	trial	go
340.9269	0.25	30	go
341.1769	0.33	11	go
341.4953	0.00	1	go
341.5019	0.50	40	
342.0019	1.50	trial	stop
342.0019	0.25	30	stop
342.2519	0.15	212	stop
342.4019	1.10	222	stop
343.5019	0.02	70	
360.7230	0.00	3	
360.7272	2.01	80	
...	...	...	
374.7858	2.01	80	
378.3026	0.50	40	
378.8026	0.61	trial	go
378.8026	0.25	30	go
379.0526	0.36	12	go
379.4028	0.00	2	go
379.4109	0.50	40	
379.9109	0.56	trial	go
379.9109	0.25	30	go
380.1609	0.31	12	go
380.4648	0.00	2	go
380.4693	0.50	40	

Fig. 3 Events in original SOCCER events. Events are represented with simple code, but specific sequences represent structural elements of the experiments such as trials. Based on the sequence of events within a trial, we can determine what the trial type is. To appropriately associate context with entire trials, instead of only the events within, one can add trial events. (This Figure has been designed using resources from Flaticon.com)

disambiguate event marker codes. The tools introduced later in this chapter allow users to specify in a JSON text file how to insert various structure markers into the event files and to subsequently disambiguate codes or to transform between event-level and trial-level encoding with no or little additional programming.

2.4.6 Missing Events and Event Values

Events that are not encoded will only contribute to implicit baseline for the signal but cannot be modeled explicitly. Visual or auditory cues indicating that participants should get ready for the trial start, as well as feedback events and other sensory cues that occur during a trial or at the beginning and ending of a block, are often omitted. Often these omitted events are seen as trivial, or they are simply not the object of inquiry for the curating researchers, yet their inclusion could result in a better fitting model.

Another problem with missing information is that values representing expected conditions can be missing entirely from a dataset because they were not recorded—or more frequently because the experimental participant failed to follow the protocol. This can cause serious problems for downstream analysis. Sometimes these types of omissions make it difficult to determine exactly where a trial or block ends when translating individual log entries to a single event marker representing the whole trial. For event-level encodings including a trial number column can be very helpful for

downstream analyses, so that specific code to check for all the potential omissions does not have to be written.

AOMIC-PIOP2

The reference paper for AOMIC-POIP2 mentions an additional feedback trial of 2000 ms if the participant responded too slowly. Figure 2 of Snoek et al. (2021) [8] shows these feedback trials as the presentation of the words "Too slow!" on the screen, but these trials did not appear in the data. The paper indicates regular trials lasted 4000 ms and that trials were preceded by a jitter interval of 0, 500, 1000 or 1500 ms. The experiment on which this is based [17] stated that a fixation cross was presented during this interval, but no mention was made in Snoek et al. (2021) [8] about whether this was the case for AOMIC-PIOP2, and no fixation markers appeared in the event file. Without access to the experimental logs, users of the data cannot tell whether these markers were omitted as unimportant or not included at all in the experimental protocol.

Another issue arises when the data file is missing expected trials of a particular type or has an unexpected distribution of trials in a particular condition. For example, some event files in AOMIC-PIOP2 contained no successful stops, and some event files contained no unsuccessful stops, although the experimental goal was to adjust the *stop_signal_delay* to achieve a 50% balance between successful and unsuccessful stops. This missing data caused crashes downstream in the *FitLins* processing when the *trial_type* column was converted to factors for modeling and a factor value previously encountered during processing other files (e.g., a successful stop or an unsuccessful stop) was found to be missing from an event file. While a large number of runs in this dataset achieved a reasonable balance of successful versus unsuccessful stop trials, the burden is on the analyst to check the event file contents and only use runs and trials appropriate for the analysis.

SOCCER

The coding of events in the SOCCER presentation log was not complete. Because the user controls which events created are pushed to the log file, it is possible to present stimuli or acquire responses without logging any information. In this case, there was a short "get ready" message before the start of each experimental block that was not pushed into the log file. Often, events are unreported because they are not the object of inquiry in the study. In this case the event may be viewed as trivial, but the event is the first after a 16-s period of rest and meant to prepare the participants for a new period of activity. Based on the experiment coding we can make a reasonable estimate of these event onsets and add them to the event file.

Another situation in the SOCCER study requiring special handling were rest blocks of 16 s between the experimental blocks. During these rest blocks an empty white screen was presented

continuously. In the presentation software logs, this presentation was coded as 8 repeat events, each lasting 2 s. If these rest events were factored for the model, there would be 8 onsets and durations. In case of an event-related analysis all these onsets would be modeled with impulse function individually. Note that HRFs interact nonlinearly when events quickly succeed each other [18]. Modeling multiple quick successive events where there are none creates an inappropriate model for the BOLD response and will likely negatively affect the results. If there are no new onsets there should be no new events, even if the experiment software internally refreshes the presentation of such an event. To analyze resting blocks a researcher would have to fold in these events with appropriate duration before analyzing based on the event file.

2.4.7 Ambiguous Encoding

Event meanings are encoded in event files using custom labels. Many downstream analyses are predicated on using these labels to define factors or contrasts for analysis. If labels (e.g., values in a *trial_type* or *code* column) are ambiguously encoded, downstream analysts will need to disambiguate them before starting analysis, usually by writing special-purpose code. Events can be ambiguous in multiple ways. Often, ambiguous encoding comes down to a lack of contextual information. The event itself could be a simple button press, but the events occurring before generally determine whether this button press was appropriate or a correct solution to the task given to the participant. Another common case is multiple button presses due to participant errors. Here, we go over some of the ambiguous codes in the example datasets.

AOMIC-PIOP2

The three *trial_type* values were: *go*, *succesful_stop* (sic), and *unsuccesful_stop* (sic). However, this did not completely encode all possible trial types, because in some cases the participant missed pressing a button entirely during a *go* trial. Because this case wasn't encoded as a separate *trial_type*, analysts downstream must write code to exclude these trials from comparisons of the go condition with stop condition. In addition to missed *go* trials, there were trials in which the participant pressed the wrong button during discrimination.

SOCCER

The SOCCER study consisted of multiple blocks, including one survey block at the end of the experiment in which the participants responded to statements on their task strategies. Throughout the entire run participants used a single button box. During experimental blocks participants used the left and right button to indicate whether there was a left or right arrow. During the survey blocks these same buttons were used to move a slider left and right. To analyze the neural correlates of button presses during experimental trials, button presses during experimental trials must be made distinct from button presses during the survey block.

Notes

1. *Ideally a trial type (or event code) should have unique codes for each possible type of trial (or event). Unambiguous encoding enables correct downstream splitting of type into factors for modeling without specialized programming.*
2. *Careful summarization of values and combinations of values in event files should be done before setting up statistical models. Appropriate rows should be dropped and missing or unusual column values handled.*
3. *All the events in a trial should be reported, ideally using event-level rather than trial-level encoding to facilitate correct automated downstream processing.*
4. *Context should be actively associated with all events to allow differentiate between identical events that have different implications for participant cognition.*

As shown in Fig. 1 on the lifecycle of data and further illustrated by the event encoding issues discussed in this section, analysis and modeling are iterative processes. No data curator, no matter how conscientious or proficient, can anticipate all of the possible questions that might be pursued by downstream analysts. Further, even when the representation of events in a data set is relatively complete, these encodings are likely to be incompatible when organized for larger analyses across multiple datasets. In the next section we discuss the modeling infrastructure represented by BIDS Stats Models and introduce tools for restructuring events without re-coding.

3 Modeling and Event Structure

In this section we give an overview of the BIDS Stats Model format and discuss the relationship of models to the event structure. We then introduce event file restructuring tools designed to help researchers restructure their event files to address particular questions.

3.1 The BIDS Stats Model Framework

Our example analyses use the *FitLins* linear modeling package to illustrate the interaction between modeling and event organization. *FitLins* is designed for large-scale automated processing using containers. Installation and examples of running *FitLins* for the examples discussed in this paper are contained in the supplementary materials at https://osf.io/93km8/. An important aspect of

Fig. 4 An example of a BIDS Stats Model for comparing responses between left and right hands. On the left: the JSON model file used as input for *FitLins*. On the right: the execution graph for the computational model

FitLins for reproducibility is its use of model specifications, which are then included with the output to fully document the computation and the results. The *FitLins* model specifications use the newly-standardized BIDS Stats Model format, an example is shown in Fig. 4.

A BIDS Stats Model encodes a multilevel hierarchical statistical model, where statistics for individual runs are combined at higher levels to obtain group statistics representative of the entire dataset or across multiple datasets. The purpose of these models is to provide a complete, re-executable record of the computations.

BIDS Stats Models have only recently been incorporated into BIDS, and while currently focused mainly on fMRI and hierarchical modeling, will likely be extended in the future to support other imaging modalities and analysis techniques. A nice introduction to BIDS Stats Models is available at [https://bids-standard.github.io/stats-models/index.html].

3.2 BIDS Stats Model Graphs

A BIDS Stats Model is represented by a JSON file (Fig. 4, left) specifying the nodes (computations) and edges (input/output relationships between nodes) of a computational graph. The graph in Fig. 4 indicates that after the computational block *Subject-level* is executed, its output is fed into two additional computational blocks, *F-test-left-right* and *T-test-contrasts*. These latter blocks pool results from individual runs across the entire dataset, as indicated by the *Level* parameters of the respective blocks. The input data to the run level computational block depends on the tool, but in this case it is the fMRI imaging files. BIDS Stats Models currently focus on general linear models (GLMs), and the event files associated with the individual imaging files provide critical input for the models as shown by the expansion of the *Subject-level* node of Fig. 4 shown in Fig. 5.

The model indicates that the BOLD signal of each imaging file should be convolved at specific time markers in the data, as specified

BIDS Stats Model Node at Run Level

```
{
   "Level":"run",
   "Name":"Subject-level",
   "GroupBy":["subject"],
   "Transformations":{
      "Transformer": "pybids-transforms-v1",
      "Instructions": [
         {"Name": "Factor", "Input": ["trial_type", "response_hand"]},
         {"Name": "And", "Input": ["trial_type.go", "response_hand.left"], "Output": "go_left"},
         {"Name": "And", "Input": ["trial_type.go", "response_hand.right"], "Output": "go_right"},
         {"Name": "Convolve", "Input":["go_right", "go_left"], "Model":"spm"}
      ]
   },
   "Model":{"X":["go_right", "go_left", "trans_*", "rot_*", 1], "Type": "glm"},
   "DummyContrasts": {"Conditions": ["go_left", "go_right"], "Test":"t"},
   "Contrasts": [
      {"Name": "left_bigger_than_right", "ConditionList": ["go_left", "go_right"], "Weights":[1, -1], "Test":"t"},
      {"Name": "right_bigger_than_left", "ConditionList": ["go_left", "go_right"], "Weights":[-1, 1], "Test":"t"},
      {"Name": "go_trial_vs_baseline", "ConditionList":["go_left", "go_right"], "Weights":[0.5, 0.5], "Test":"t"},
      {"Name": "Movement-related effects", "ConditionList":,"Weights": [...],"Test": "F"}
   ]
}
```

Fig. 5 An expansion of the Subject-level node of Fig. 4

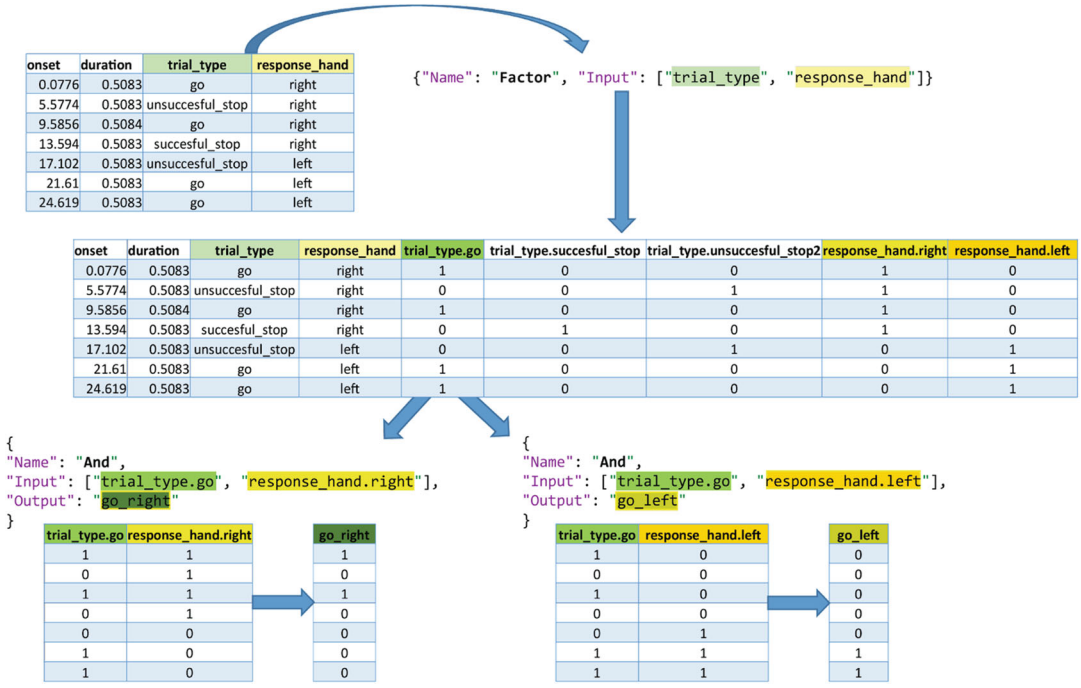
by *go_right* and *go_left*, using an HRF model based on the canonical HRF as provided in *spm* [16]. Here *go_right* and *go_left* are factor vectors of the same length as the *events.tsv* file associated with the imaging file being processed. These factor vectors have 1's in positions for events that satisfy the *go_right* and *go_left* conditions, respectively. The remaining positions are 0.

3.3 BIDS Transformations

If *go_right* and *go_left* were values in a column of the *events.tsv* files, the computation could proceed directly, but since they were not, the events file must undergo some transformations prior to the application of the statistical model. BIDS Transformations, which are under development for incorporation into the BIDS specification, perform logical, selection, and other operations on columns of an events file to generate factor vectors that can be used as input to the models.

The BIDS Transformations can be specified directly in the BIDS Stats model and Fitlins runs these internally without a requirement for additional coding. BIDS Transformations consist of a list of dictionaries, each specifying an operation. The operations are performed in succession. For the transformations in the example of Fig. 5, the input consists of column names from the internal representation of the events file, and the outputs are also names of derived columns computed from the same internal event file.

Figure 6 shows how this process works for the transformations specified in Fig. 5. To obtain the factor columns *go_right* and *go_left*, the *Factor Transformation* creates new factor columns from the unique values in the input columns *trial_type* and *response_hand*. These newly created columns follow the naming



onset	duration	trial_type	response_hand
0.0776	0.5083	go	right
5.5774	0.5083	unsuccesful_stop	right
9.5856	0.5084	go	right
13.594	0.5083	succesful_stop	right
17.102	0.5083	unsuccesful_stop	left
21.61	0.5083	go	left
24.619	0.5083	go	left

onset	duration	trial_type	response_hand	trial_type.go	trial_type.succesful_stop	trial_type.unsuccesful_stop2	response_hand.right	response_hand.left
0.0776	0.5083	go	right	1	0	0	1	0
5.5774	0.5083	unsuccesful_stop	right	0	0	1	1	0
9.5856	0.5084	go	right	1	0	0	1	0
13.594	0.5083	succesful_stop	right	0	1	0	1	0
17.102	0.5083	unsuccesful_stop	left	0	0	1	0	1
21.61	0.5083	go	left	1	0	0	0	1
24.619	0.5083	go	left	1	0	0	0	1

trial_type.go	response_hand.right	go_right
1	1	1
0	1	0
1	1	1
0	1	0
0	0	0
1	0	0
1	0	0

trial_type.go	response_hand.left	go_left
1	0	0
0	0	0
1	0	0
0	0	0
0	1	0
1	1	1
1	1	1

Fig. 6 The process of using BIDS Transformations to create factor vectors out of event file columns

convention *column_name.column_value*, so the factor column representing *go* trials is called *trial_type.go*. Since we only want to analyze *go* trials, we apply the **And** transformation to the factor vectors for *trial_type.go* and *response_hand.right* to create a new column, we have chosen to call *go_right*. A similar operation is specified for the left hand responses. These factors, which never actually appear in the events file itself, are used as input for subsequent computations.

The BIDS Transformations are necessary because the event file columns often do not directly correspond to the needed factors. They promote reproducibility because they are incorporated as part of the model itself. However, there are a limited number of transformations available, and the results are sometimes hard to debug since the actual event file that goes into the computation is not directly visible (though it is possible to include something to dump the internal event files as part of the computation). For further information BIDS Stats Models (BIDS Extension Proposal 2) see:

https://docs.google.com/document/d/1bq5eNDHTb6Nkx3WUiOBgKvLNnaa5OMcGtD0AZ9yms2M/edit and on the BIDS Transformations specification:

https://docs.google.com/document/d/1uxN6vPWbC7ciAx2XWtT5Y-lBrdckZKpPdNUNpwRxHoU/edit#heading=h.kuzdziksbkpm).

The next section introduces an alternative approach that can be used instead of or in addition to BIDS Transformations.

3.4 Event File Transformations

Previous subsections presented an overview of the modeling process and showed that, at least for linear modeling, the initial analysis relies on the specification of factor vectors reflecting the aspects of the data to be modeled. In most cases, the original event files will not have those factors directly present, and they must be derived from the information available depending on the requirements of the particular model. BIDS Transformations play an important role in deriving suitable factor vectors, but the results of these transformations are kept internally, and data is often not perfectly aligned with the required structure.

We have developed an external event transformation mechanism (event remodeling), which is also based on specifying transformations using JSON files for reproducibility. This section introduces these transformations and demonstrates their usage. These transformations can be used during analysis, but also can be used for permanently restructuring event files during the curation of files. The transformations are not currently part of the BIDS Transformations specification, but we hope some of the event remodeling operations will eventually become part of this specification.

The event transformation strategy relies on creating backup event files as shown in Fig. 7.

The process begins by creating a backup copy of the original events files. This backup remains the same throughout the process. When a transformation is to be done, remodeling always makes a clean copy of the events file from the backup. Thus, the remodeling should always assume that it is starting with the original events file. This assures a consistent state of the data throughout the process. Remodeling takes the newly copied *events.tsv* file and the JSON specification file, performs the specified transformations, and then rewrites the *events.tsv* file.

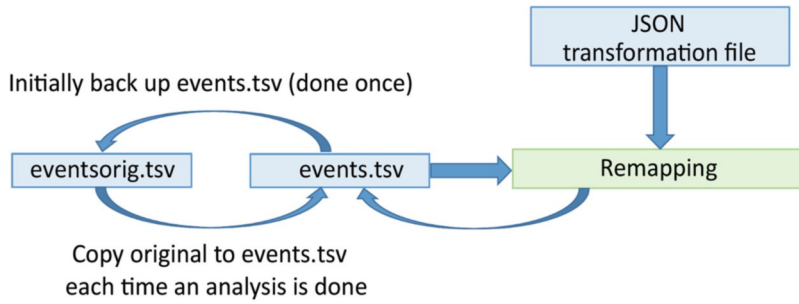


Fig. 7 The process of event file remodeling

Note
After transformation, users can review their files and use the summary tools to make sure that the transformations have the expected results. This review is particularly important in practice as most data has some unexpected quirks such as missing data, missing events as described above. Once satisfied with the results, researchers can perform analyses using tools such as *FitLins* as though the event files were the originals.

3.5 Tools for Restructuring Events

We have developed some event file restructuring tools patterned after the BIDS Transformations. Like BIDS Transformations and BIDS Stats Models, these event remapping tools use a JSON specification file to list the transformations to be performed in succession on the rows and/or columns of every *events.tsv* file in a BIDS dataset. Unlike the BIDS Transformations, however, these operations save the transformed files, so that they can be verified manually and by summarization tools.

The top-level structure of the remodeling file is a list of transformations rather than a dictionary, because a transformation of a particular type may occur at multiple stages during the remapping process. The transformations are performed in order.

Each transformation is represented as a dictionary with keys: *operation*, *description*, and *parameters*. Table 3 summarizes the transformations that are currently available. The toolbox is part of a larger toolset that supports event handling and annotation using the HED (Hierarchical Event Descriptors) framework. A more detailed listing of the event file remodeling operations and their parameters along with tutorials can be found at the File Remodeling Quickstart tutorial [https://www.hed-resources.org/en/latest/HedRemodelingQuickstart.html] and the File Remodeling Tools documentation [https://www.hed-resources.org/en/latest/HedRemodelingTools.html].

Figure 8 shows a simple example of an event restructuring specification that deletes the *sample* and *value* columns and then reorders the columns so the *onset*, *duration*, *event_type*, and *task_role* columns are the first four columns. Since *keep_others* is true, other columns are placed at the end.

When restructuring is performed using the HED remodeling tools with this file and the BIDS root directory as input, these operations will be performed on every *events.tsv* file in the BIDS dataset.

Table 3
Some standard event transformations available in the HED remodeling tools

Operation	Purpose
factor_column	Produce factor columns based on presence or absence of specified values in a column.
factor_hed_tags	Produce factor columns based on a HED tag search string.
factor_hed_type	Produce factor columns from HED type (e.g., *condition-variable* creates factor columns based on the annotated experimental design).
merge_consecutive	Merges several consecutive events of the same type into one event with duration being the span of the merged events.
remap_columns	Map the values of *n* columns into new values in *m* columns using a dictionary lookup.
remove_columns	Remove the specified columns if present.
remove_rows	Remove rows where specified columns take particular values.
rename_columns	Rename columns by providing old names and new names.
reorder_columns	Reorder the columns in the specified order. Columns not included are discarded or placed at the end.
split_rows	Split specified rows into multiple rows in the event file and adjust the meanings of the columns—Usually for unfolding trials into individual events.

```
[
  {
    "operation": "remove_columns",
    "description": "Get rid of the sample and the value columns.",
    "parameters": {
      "remove_names": ["sample", "value"],
      "ignore_missing": true
    }
  },
  {
    "operation": "reorder_columns",
    "description": "Want event_type and task_role columns after onset and duration.",
    "parameters": {
      "column_order": ["onset", "duration", "event_type", "task_role"],
      "ignore_missing": false,
      "keep_others": true
    }
  }
]
```

Fig. 8 A JSON specification for event file restructuring using the HED remodeling tools

4 Example Data Analysis

In this section we present the results of two contrasts on the AOMIC-PIOP2 and SOCCER data: *successful stop* (sic) versus *unsuccessful stop* (sic) and *left* versus *right*. Before we are able to run these contrasts we must resolve some of the problems we described earlier, mainly this considers the issue of trial-level encoding versus event-level encoding. We consider simple *t*-statistic maps [*see* Glossary] as outcomes.

A few notes should be given on the interpretation of these maps, which are created by applying statistical tests to individual voxels in the fMRI images. The process of deriving whole brain fMRI results is strongly influenced by thresholding procedures. Many strategies have been developed to deal with the massive multiple comparison problem, without applying the most conservative Bonferroni approaches, such as calculating the Family Wise Error, or the False Discovery Rate [19]. Often these approaches are combined with some cluster thresholding, meaning we require a specified number of voxels to pass the threshold before we accept the result as significant. When looking at the basic *t*-statistic maps we cannot say definitively whether higher values are a reflection of true differences between conditions. Here we broadly view patterns across datasets to showcase which differences could be points of interest.

The Nilearn python library [https://nistats.github.io/] [*see* Resources] has several functions for applying statistical thresholds to data, including FDR and cluster thresholds. We have included a simple niistats script that can be used to load in the unthresholded statistical maps created by *FitLins* and apply some basic thresholds in the supplementary material. More information on this process can be found here: [https://nilearn.github.io/stable/auto_examples/05_glm_second_level/plot_thresholding.html].

All files necessary to run the *FitLins* analysis for both datasets, as well as all *FitLins* results can be found in the supplementary materials [https://osf.io/93km8/]. In the case of the AOMIC-PIOP2 dataset, the remapping and model files can be applied to the entire dataset as found on OpenNeuro, to obtain results for the full 200 participants if desired.

4.1 Successful Versus Unsuccessful Stops

The primary focus of stop-signal tasks is to study the interaction of response and inhibition. Because the task load and task type were very different in the two datasets, one would expect some differences in the interplay of response and inhibition in the two cases. However, there might be some overlap in regions related to inhibition processes themselves. As an example comparison, we choose successful stop versus unsuccessful stop trials for the two datasets. We use the onset of the go signal as the marker for these events in

both datasets. We use the encoding of successful and unsuccessful stop trials from the AOMIC dataset without event remodeling. The models for this example are available in the supplemental materials [https://osf.io/93km8/].

Figure 9 shows results for the contrast of successful versus unsuccessful stops in the two datasets. As expected the results from this smaller sample (top two graphs) show limited convergence of regional activation across participants. Interpretation here is difficult because of the small sample size. However, the larger AOMIC-PIOP2 dataset shows distinctive patterns which require additional thresholding.

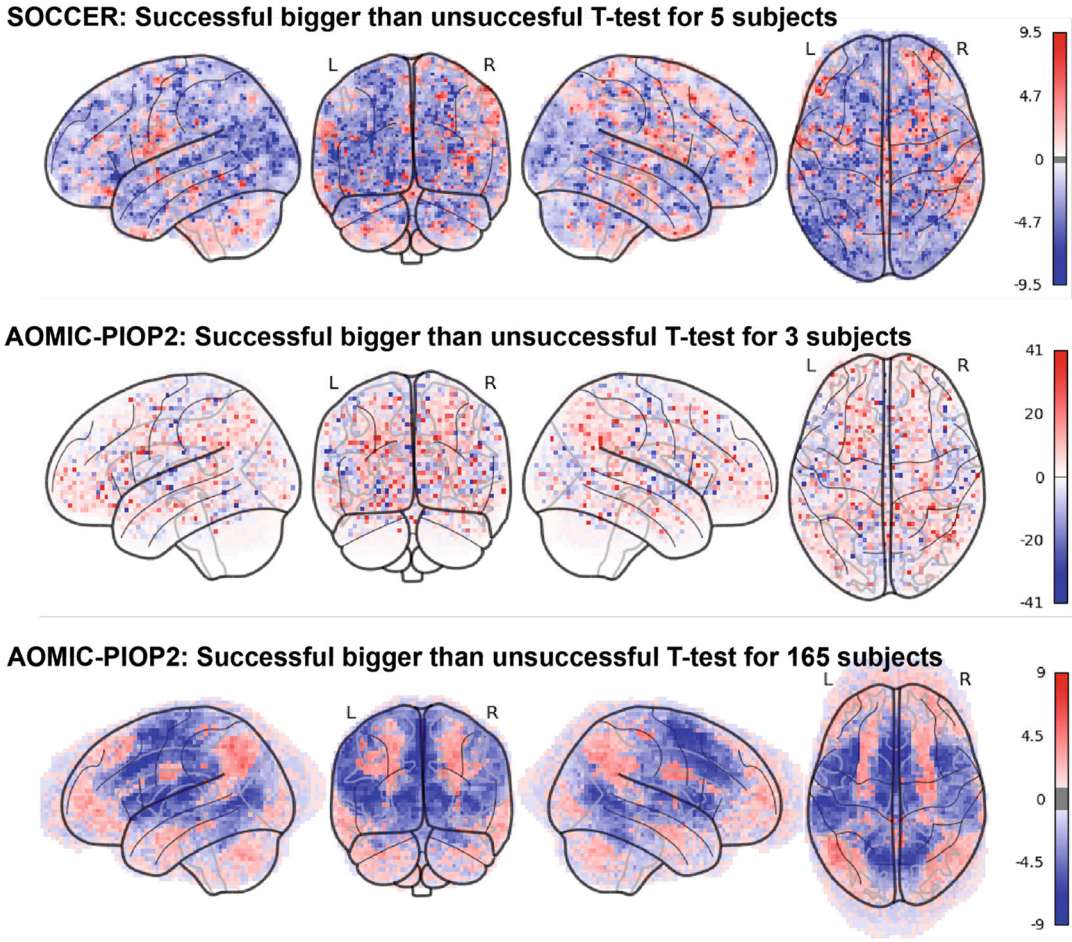


Fig. 9 *T*-test comparison of regions where response for successful stop trials was greater than for unsuccessful stop trials. Top graph: SOCCER for the five-subject demo data. Middle graph: AOMIC-PIOP2 for the three-subject demo data. Bottom graph: AOMIC-PIOP2 for 165 subjects. Note: three subjects in AOMIC-PIOP2 had no successful stops and one subject had no unsuccessful stops. In all, 61 subjects had fewer than ten trials of one of the types and were removed. Color scales represent *t*-values

Based on the *t*-statistic maps of the all participants we see a larger difference to the successful versus unsuccessful trials in frontal cortex, cingulate gyrus, as well as inferior parietal gyrus and regions around the supramarginal gyrus. Some of these regions have been found previously in studies related to response inhibition. A meta-analysis into different inhibition tasks found peak activations across several stop signal tasks in the right middle cingulate gyrus, bilateral supramarginal gyrus, and right inferior parietal gyrus, as well as several other regions that do not show a clear difference here [20]. By comparing results with other types of inhibition tasks they found response inhibition was most likely controlled by a fronto-parietal network and ventral network, which is congruent with the pattern we see.

4.2 Left Versus Right Responses

Both datasets had participants press left versus right buttons to indicate their responses in the experiment's discrimination task. While AOMIC-PIOP2 experimental setup used separate fiber-optic response pads with four buttons for each hand, SOCCER only used a single response pad, located at the participant's right hand. Participants in the SOCCER dataset were instructed to use their right index finger and ring finger for responses, while participants in the AOMIC-PIOP2 study used their left and right hand index fingers to press buttons on the left and right response pads. Based on this configuration, we expect a large difference in neural activation between left and right motor cortex for left and right button presses in the AOMIC-PIOP2 results, while we expect no such differences for the SOCCER data.

Because SOCCER uses event-level encoding, the button presses occupy their own rows in the event tables with onsets corresponding to the times of the button presses. In contrast, AOMIC-PIOP2 uses trial-level encoding, and the onset of the trial is the presentation of the face image, not the time of the button press. One possible model for the AOMIC-PIOP2 left versus right comparison was introduced in Figs. 5 and 6 using the BIDS Transformations. The difficulty with using this model for comparison of left-right responses with SOCCER is that SOCCER encodes the response events individually and the times of these events are recorded as the times of the button presses rather than the times of the image presentations. The AOMIC-PIOP2 response times average 1.02 s, but there is significant variability among subjects and trials. An alternative is to recode the AOMIC-PIOP2 event files to more closely match the response events of SOCCER. In the next section we show how to do this re-coding using the remodeling tools.

4.2.1 AOMIC-PIOP2 Event Preparation

Figure 10 shows the remodeling steps required. The goal is to produce a new event file where the rows represent the times of the button presses in response to the face images. Since the new

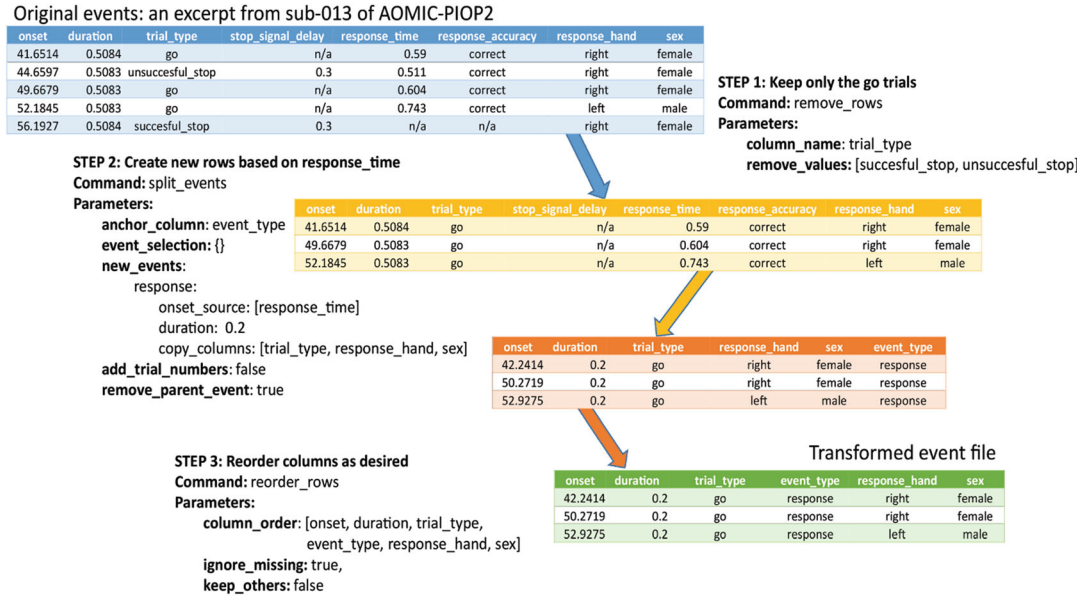


Fig. 10 Remodeling AOMIC-PIOP2 response events. The *response_time* events are transformed into new events rows, with the original events being removed

response times will be the original *onset* plus the *response_time*, we first remove any rows where the *response_time* is not defined.

The next step is to create new events from the original trial event. Since *remove_trial_parent* is true, the original trial event will not appear in the remodeled file. Any number of new events can be created during a *split_event*, but the example only shows creation of *response* events. The new_column parameter designates the column in which the "codes" representing these new events are recorded. Here the only code is *response*. The *onset_source* designates a list of values (and/or column names from which to extract the values for each event) that are added to the onset value to produce the onset of the new event. Some values in the list could be negative. The created events don't have to be created in order of their onsets as the onsets are resorted at the end of the split_event process.

Figure 11 shows the remodeling transformation JSON file that performs these operations. The file consists of a list of four dictionaries corresponding to the three steps in transforming the event file. The first two transformations focus on go trials and make sure that all response times are defined. The next transformation creates a new event for each go trial with onset at the time of button press. The final transformation reorders the columns as required. Additional information and tutorials on using the remodeling facilities can be found in the File Remodeling tools documentation: [https://www.hed-resources.org/en/latest/HedRemodelingTools.html].

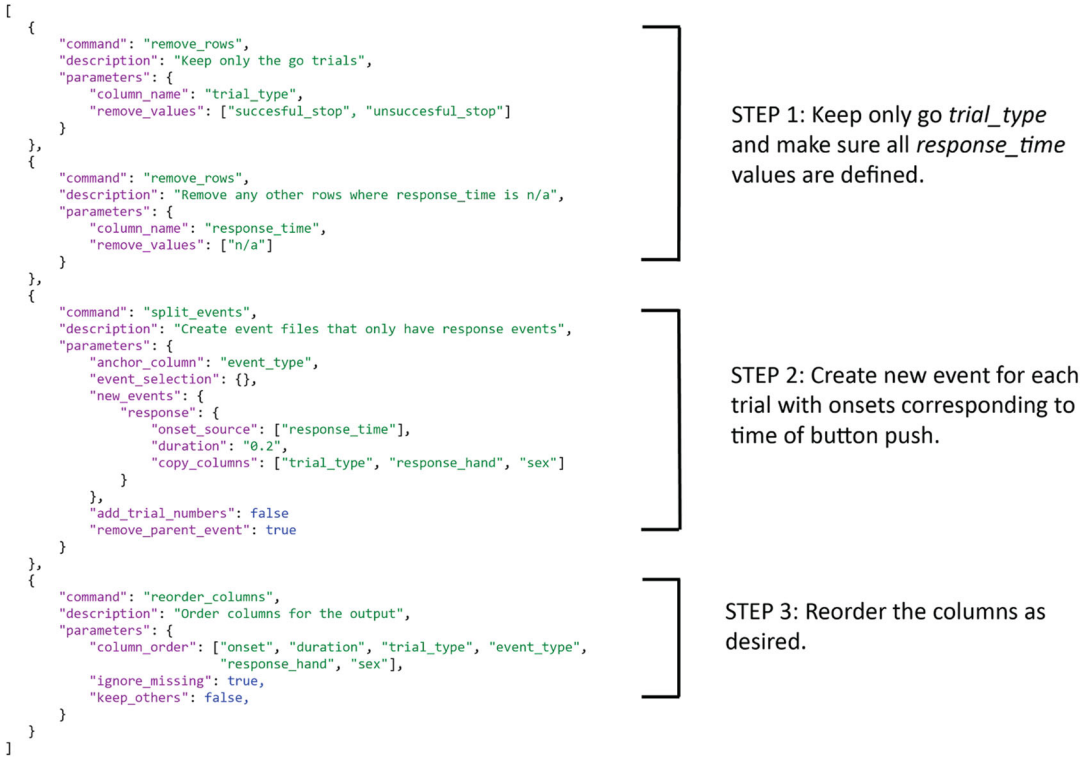


Fig. 11 JSON file with instructions for restructuring AOMIC-PIOP2 event files for Model 2 above

The script of Fig. 11 includes *remove_rows* operations for both *response_time* equal to *n/a* and for trial_type values of either *succesful_stop* or *unsuccesful_stop* to assure that only successful *go* trials with *response_time* defined are included. For simplicity of explanation, we did not also include a *remove_rows* operation for *response_accuracy* values of *incorrect*, as there were relatively few trials where this occurred. However, it is a good idea to consider all unusual cases in developing a remodeling script.

4.2.2 SOCCER Event Preparation

To run a comparable contrast on the SOCCER data, two issues needed to be addressed: resolution of ambiguity in the button press codes and detection of invalid trials. The first issue illustrates inserting block markers, the second issue illustrates insertion of trial structure.

As described earlier (Subheading "SOCCER"), the SOCCER data contained mainly event level information. Left and right button presses were coded as either code 1 or code 2 in the log file. But these same buttons were used to answer questions during the survey block, as well as to perform the experiment task. Because of this, button presses during experimental trials must be distinguished from button presses in the survey trials. To address this ambiguity, we select all events between code "40" and "80" and

define them as part of the experiment block group. Once blocks have been distinguished, the unique codes for the survey blocks may be substituted to distinguish from the experimental blocks.

The invalid trial problem illustrates the use of the second approach based on labeling specific event sequences. We are interested in the left versus right responses during *go* trials. *Go* trials can be identified based on specific sequences within trial groups. Once we have identified the trial groups, we provide individual labels for each sequence in the group.

Table 4 shows an overview of all the sequences found in trial groups labeled as go trials, successful stop trials, or unsuccessful stop trials. These labeled sequences can be used not only for labeling trial types, but also for providing detailed summaries of the trials that occurred in the dataset.

Some sequences in the table are more difficult to label such as trial groups only containing a single participant button press event. We consider these trials invalid. These responses were either corrections from incorrectly solved *go* trials, or double responses. The participant pressed the button before the stop signal was presented in some trials. In some cases, the button presses were so early that the stop signal was never even presented. We classified these latter trials as *go* trials as this is what it would be perceived from the participant perspective.

Table 4
Trial sequences in SOCCER dataset along with appropriate trial labels

Sequence	Trial category
30, 11, 1	Go
30, 12, 2	Go
30, 12, 1	Go
30, 211, 221	Successful stop
30, 212, 222	Successful stop
30, 211, 221, 1	Unsuccessful stop
30, 212, 222, 2	Unsuccessful stop
30, 211, 221, 2	Unsuccessful stop
30, 211, 1	Go
30, 212, 2	Go
30, 211, 2	Go
30, 211, 1, 221	Premature signal response
1	Invalid
2	Invalid

The details of these transformations are complex and are beyond the scope of this chapter. For more information, see [https://www.hed-resources.org/en/latest/HedRemodelingTools.html].

4.2.3 Left Versus Right Results

Figure 12 shows a comparison of the unthresholded *FitLins* statistical maps for the *T*-test for go trials where activation for the left button press is greater than for the right button press. As expected, the SOCCER demo data (top graph) does not show an obvious difference across hemispheres.

On the other hand, even in a dataset as small as the three-subject demo dataset, AOMIC-PIOP2 does show a visual

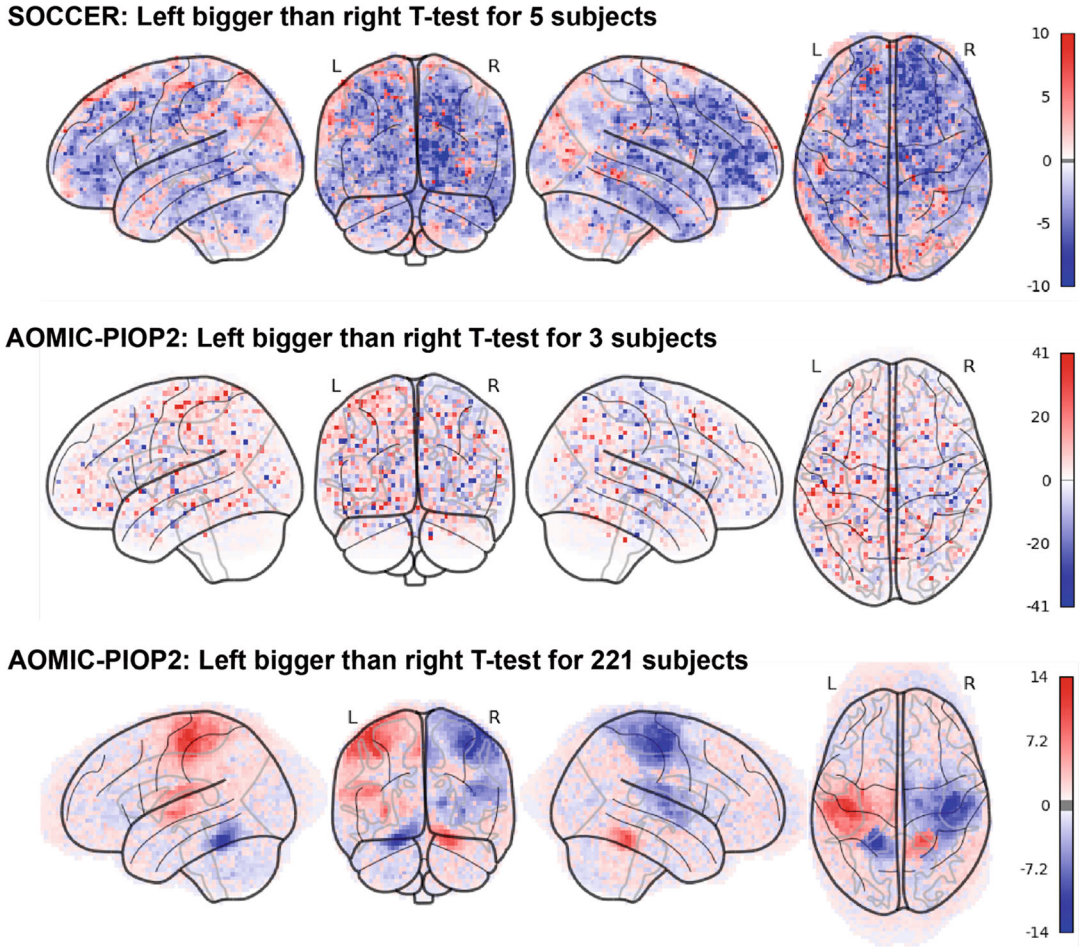


Fig. 12 *T*-test comparison of regions where response for left button press is greater than for right button presses in go trials. Top graph: SOCCER for the five-subject demo data. Middle graph: AOMIC-PIOP2 for the three-subject demo data. Bottom graph: AOMIC-PIOP2 for 221 subjects. Note: three subjects in AOMIC-PIOP2 had no *go* trial left button presses and two had no *go* trial right button presses and were excluded. Color scales represent *t*-values

difference between hemispheres for left larger than right button presses (middle graph). The distinction is much more clearly demonstrated in the 221-subject AOMIC-PIOP2 dataset (bottom graph). Unfortunately, the difference is in the opposite direction of our expectations—with left hand actions associated with higher *t*-values in the left hemisphere. We also see the contralateral (normally ipsilateral) activation in the cerebellum. This discrepancy could be a mistake in the model, a reversal of designations in the event files, or even be a reversal of the hemispheres during image processing. Since the original log files are not available we have not been able to rule out any of these possibilities. However, this result demonstrates the usefulness of running even relatively trivial comparisons to cross-check the consistency of the data and results. All of the model files and results are available in the supplementary materials.

5 Discussion and Conclusion

The stop signal task used as an example throughout this chapter is a well-established paradigm for studying the mental processes *response initiation* and *inhibition*. We have used two datasets to showcase the process of event restructuring and annotation for comparable analysis. Our two demo datasets illustrate two common use cases: downloading datasets available on open repositories (AOMIC-PIOP2) and structuring local data in standardized formats (SOCCER). Based on the limited sample sizes of the demo datasets, we cannot draw conclusions about regions related to response inhibition across studies, rather our purpose was to illustrate event restructuring issues and approaches.

We illustrated the issues using two comparison problems: successful versus unsuccessful stops and left versus right responses. In the first example, there are comparable events in the two datasets, and the standard BIDS Transformation mechanisms are sufficient for modeling. The left versus right comparison is more complicated because the event encoding of the two datasets prevents direct comparison of these events. We use this example as an illustration of the process of event remodeling to transform the event files of AOMIC-PIOP2 for a better comparison with SOCCER.

Events play an essential role in neuroimaging data management, from initial preparation using the experimental logs to advanced task-based analysis. Creating appropriate event representations for a given application often requires considerable time investment, and the problem is compounded when the analysis includes datasets from different experiments and laboratories as demonstrated by the comparison of the AOMIC-PIOP2 and SOCCER datasets presented in this chapter.

Event files also play an important role in reproducible and transparent analyses. Transparency requires thorough documentation of events and use of standardized terms for describing cognitive tasks and their implied role in cognition. Careful documentation of control variables and specific targeting of mental concepts is an important part of the practice of cognitive science. More recent incorporations of naturalistic paradigms such as movie watching also depend on careful structuring and flexible annotation of events occurring during the experiment.

To address these issues, we have developed a framework and event restructuring tools (remodeler) that allow users to modify a dataset's event files by specifying a series of operations in a JSON text file. Using such a file not only allows researchers to avoid writing one-off code for each analysis, but also results in a file that clearly documents the operations performed on the events files so that they can be easily reproduced. These restructuring tools are well-integrated with BIDS and augment the BIDS Stats Model and BIDS Transformations. The tools are also integrated with HED (Hierarchical Event Descriptors) [5] for more advanced analysis and standardized annotation of events. Event restructuring, as well as the role of HED in restructuring, is discussed in [https://www.hed-resources.org/en/latest/HedRemodelingQuickstart.html] and [https://www.hed-resources.org/en/latest/HedRemodelingTools.html].

Aggregating neuroimaging data is an important aspect of verifying neuroimaging results from often small sample sizes, and establishing consistent results across studies with often variable designs and participant groups. Besides questions about the generalizable results, however, there are also questions in neuroscience about the effect of variations in experimental design on neural activation. In order to learn more about this, it is important to compare experiments on an event level. Data sharing has also proven useful for method testing and validation as well as for the development and testing of software [6].

Event representation for documentation and analysis rarely receives the attention it deserves, given its complexity and how essential it is to a valid analysis. We have shown that synchronization of multiple datasets to allow for comparable analysis results requires a significant investment of time and resources, even for relatively straightforward stimulus-response paradigms. A deep understanding of the executed paradigms and the events remains essential. However, with better guidelines and more tools for event handling, event processing can be made less time intensive for users of shared data as well as for those collecting new data.

Author Contributions

KR and MD conceived software and demonstration and drafted the manuscript. FR and JB were responsible for data collection in the SOCCER study. MP worked on data curation for SOCCER study. All authors contributed useful comments to tutorial conception and the finalization of the manuscript.

Funding

This study was funded by NIMH grant R01MH126700-01A1 and by a grant to the first and fifth authors (MJMD, ANR) from the Doctoral College "Imaging the Mind" [FWF W 1233-B].

References

1. (2017) E-Prime® | Psychology Software Tools. In: Psychology Software Tools | Solutions for Research, Assessment, and Education. https://pstnet.com/products/e-prime/. Accessed 5 Mar 2024
2. Peirce J, Gray JR, Simpson S, MacAskill M, Höchenberger R, Sogo H, Kastman E, Lindeløv JK (2019) PsychoPy2: experiments in behavior made easy. Behav Res Methods 51: 195–203. https://doi.org/10.3758/s13428-018-01193-y
3. Gorgolewski KJ, Auer T, Calhoun VD, Craddock RC, Das S, Duff EP, Flandin G, Ghosh SS, Glatard T, Halchenko YO, Handwerker DA, Hanke M, Keator D, Li X, Michael Z, Maumet C, Nichols BN, Nichols TE, Pellman J, Poline J-B, Rokem A, Schaefer G, Sochat V, Triplett W, Turner JA, Varoquaux G, Poldrack RA (2016) The brain imaging data structure, a format for organizing and describing outputs of neuroimaging experiments. Sci Data 3:160044. https://doi.org/10.1038/sdata.2016.44
4. Robbins K, Truong D, Appelhoff S, Delorme A, Makeig S (2021) Capturing the nature of events and event context using hierarchical event descriptors (HED). NeuroImage 245:118766. https://doi.org/10.1016/j.neuroimage.2021.118766
5. Robbins K, Truong D, Jones A, Callanan I, Makeig S (2022) Building FAIR functionality: annotating events in time series data using hierarchical event descriptors (HED). Neuroinformatics 20:463–481. https://doi.org/10.1007/s12021-021-09537-4
6. Markiewicz CJ, Gorgolewski KJ, Feingold F, Blair R, Halchenko YO, Miller E, Hardcastle N, Wexler J, Esteban O, Goncavles M, Jwa A, Poldrack R (2021) The OpenNeuro resource for sharing of neuroscience data. elife 10:e71774. https://doi.org/10.7554/eLife.71774
7. Gorgolewski KJ, Alfaro-Almagro F, Auer T, Bellec P, Capotă M, Chakravarty MM, Churchill NW, Cohen AL, Craddock RC, Devenyi GA, Eklund A, Esteban O, Flandin G, Ghosh SS, Guntupalli JS, Jenkinson M, Keshavan A, Kiar G, Liem F, Raamana PR, Raffelt D, Steele CJ, Quirion P-O, Smith RE, Strother SC, Varoquaux G, Wang Y, Yarkoni T, Poldrack RA (2017) BIDS apps: improving ease of use, accessibility, and reproducibility of neuroimaging data analysis methods. PLoS Comput Biol 13:e1005209. https://doi.org/10.1371/journal.pcbi.1005209
8. Snoek L, van der Miesen MM, Beemsterboer T, van der Leij A, Eigenhuis A, Steven Scholte H (2021) The Amsterdam open MRI collection, a set of multimodal MRI datasets for individual difference analyses. Sci Data 8:85. https://doi.org/10.1038/s41597-021-00870-6
9. Esteban O, Markiewicz CJ, Blair RW, Moodie CA, Isik AI, Erramuzpe A, Kent JD, Goncalves M, DuPre E, Snyder M, Oya H, Ghosh SS, Wright J, Durnez J, Poldrack RA, Gorgolewski KJ (2019) fMRIPrep: a robust preprocessing pipeline for functional MRI. Nat Methods 16:111–116. https://doi.org/10.1038/s41592-018-0235-4
10. Markiewicz CJ, De La Vega A, Wagner A, Halchenko YO, Finc K, Ciric R, Goncalves M, Nielson DM, Kent JD, Lee JA, Bansal S,

Poldrack RA, Gorgolewski KJ (2022) Poldracklab/fitlins: 0.10.1

11. Markiewicz C, Bottenhorn K, Chen G, Vega A de L, Esteban O, Maumet C, Nichols T, Poldrack R, Poline J-B, Yarkoni T (2021) BIDS Statistical Models – An implementation-independent representation of General Linear Models p. 1
12. Yarkoni T, Markiewicz CJ, de la Vega A, Gorgolewski KJ, Salo T, Halchenko YO, McNamara Q, DeStasio K, Poline J-B, Petrov D, Hayot-Sasson V, Nielson DM, Carlin J, Kiar G, Whitaker K, DuPre E, Wagner A, Tirrell LS, Jas M, Hanke M, Poldrack RA, Esteban O, Appelhoff S, Holdgraf C, Staden I, Thirion B, Kleinschmidt DF, Lee JA, di Oleggio V, Castello M, Notter MP, Blair R (2019) PyBIDS: python tools for BIDS datasets. J Open Source Softw 4:1294. https://doi.org/10.21105/joss.01294
13. Logan GD, Cowan WB (1984) On the ability to inhibit thought and action: a theory of an act of control. Psychol Rev 91:295–327. https://doi.org/10.1037/0033-295X.91.3.295
14. Ramautar JR, Slagter HA, Kok A, Ridderinkhof KR (2006) Probability effects in the stop-signal paradigm: the insula and the significance of failed inhibition. Brain Res 1105: 143–154. https://doi.org/10.1016/j.brainres.2006.02.091
15. Logothetis NK, Pauls J, Augath M, Trinath T, Oeltermann A (2001) Neurophysiological investigation of the basis of the fMRI signal. Nature 412:150–157. https://doi.org/10.1038/35084005
16. BIDS Stats Models—BIDS Stats Models Specification. https://bids-standard.github.io/stats-models/. Accessed 5 Mar 2024
17. Jahfari S, Waldorp L, Ridderinkhof KR, Scholte HS (2015) Visual information shapes the dynamics of corticobasal ganglia pathways during response selection and inhibition. J Cogn Neurosci 27:1344–1359. https://doi.org/10.1162/jocn_a_00792
18. Friston KJ, Josephs O, Rees G, Turner R (1998) Nonlinear event-related responses in fMRI. Magn Reson Med 39:41–52. https://doi.org/10.1002/mrm.1910390109
19. Logan BR, Rowe DB (2004) An evaluation of thresholding techniques in fMRI analysis. NeuroImage 22:95–108. https://doi.org/10.1016/j.neuroimage.2003.12.047
20. Zhang R, Geng X, Lee TMC (2017) Large-scale functional neural network correlates of response inhibition: an fMRI meta-analysis. Brain Struct Funct 222:3973–3990. https://doi.org/10.1007/s00429-017-1443-x