Tags: tutorial meg raw preprocessing meg-language

Trigger-based trial selection

Introduction

This tutorial describes how to define epochs-of-interest (trials) from your recorded MEG-data, and how to apply the different preprocessing steps. This tutorial does not show yet how to analyze (e.g. average) your data.

If you are interested in how to do preprocessing on your data prior to segmenting it into trials, you can check the Preprocessing - Reading continuous data tutorial. There, you can also find information about how to preprocess EEG data. If you want to learn how to segment EEG data into trials, check the tutorial on Preprocessing of EEG data and computing ERPs.

Background

In FieldTrip the preprocessing of data refers to the reading of the data, segmenting the data around interesting events such as triggers, temporal filtering and optionally rereferencing. The **ft_preprocessing** function takes care of all these steps, i.e., it reads the data and applies the preprocessing options.

There are largely two alternative approaches for preprocessing, which especially differ in the amount of memory required. The first approach is to read all data from the file into memory, apply filters, and subsequently cut the data into interesting segments. The second approach is to first identify the interesting segments, read those segments from the data file and apply the filters to those segments only. The remainder of this tutorial explains the second approach, as that is the most appropriate for large data sets such as the MEG data used in this tutorial. The approach for reading and filtering continuous data and segmenting afterwards is explained in another tutorial.

Preprocessing involves several steps including identifying individual trials from the dataset, filtering and artifact rejections. This tutorial covers how to identify trials using the trigger signal. Defining data segments of interest can be done

- according to a specified trigger channel
- according to your own criteria when you write your own trial function

Examples for both ways are described in this tutorial, and both ways depend on **ft_definetrial**.

The output of ft_definetrial is a configuration structure containing the field cfg.trl. This is a matrix representing the relevant parts of the raw datafile which are to be selected for further processing. Each row in the `trl` matrix represents a single epoch-of-interest, and the `trl` matrix has at least 3 columns. The first column defines (in samples) the beginpoint of each epoch with respect to how the data are stored in the raw datafile. The second column defines (in samples) the endpoint of each epoch, and the third column specifies the offset (in samples) of the first sample within each epoch with respect to timepoint 0 within that epoch.

Details of the MEG language dataset

The MEG data set used in the tutorials is from a language study on semantically congruent and incongruent sentences that is described in detail in Wang et al. (2012). Three types of sentences were used in the experiment. In the fully congruent condition (FC) the sentences ended with a high-cloze probability word, e.g. *De klimmers bereikten eindelijk de top van de berg* (*The climbers finally reached the top of the mountain*) In the fully incongruent condition (FIC) sentences ended with a semantically anomalous word which was totally unexpected given the sentential context, e.g., *De klimmers bereikten eindelijk de top van de tulp* (*The climbers finally reached the top of the tulip*). The third type of sentences ended with a semantically anomalous word that had the same initial phonemes (and lexical stress) as the high-cloze words from the congruent condition: initially congruent (IC). There were 87 trials per condition for each of the three conditions, and a set of 87 filler sentences were added. From the EEG literature it is known that a stronger negative potential is produced following incongruent compared to congruent sentence endings about 300-500 ms after the word onset. This response is termed the N400 effect[1] [2]. For more information about the materials take a look at the published EEG experiment using the same sentence materials[3].

In the study applied here, the subjects were seated in a relaxed position under the MEG helmet. Their task was to attentively listen to spoken sentences. They were informed that some of the sentences would be semantically anomalous. Acoustic transducers were used to deliver the auditory stimuli. After a 300-ms warning tone, followed by a 1200 ms pause, a sentence was presented. Every next trial began 4100 ms after the offset of the previous sentence. To reduce eye blinks and movements in the time interval in which the sentence was presented, subjects were instructed to fixate on an asterisk presented visually 1000 ms prior to the beginning of the sentence. The asterisk remained on the screen until 1600 ms after the onset of the spoken sentence. Subjects were encouraged to blink when the asterisk was not displayed on the screen.

MEG signals were recorded with a 151-channel CTF system. In addition, the EOG was recorded to later discard trials contaminated by eye movements and blinks. The ongoing MEG and EOG signals were lowpass filtered at 100 Hz, digitized at 300 Hz and stored for off-line analysis. To measure the head position with respect to the sensors, three coils were placed at anatomical landmarks of the head (nasion, left and right ear canal). While the subjects were seated under the MEG helmet, the positions of the coils were determined before and after the experiment by measuring the magnetic signals produced by currents passed through the coils.

The MEG data are stored as epochs or trials of fixed length around each stimulus trigger, i.e. the file does not represent a continuous record of the data and the data in the inter-trial-interval is not stored. The consequence is that the data cannot be represented as a continuous record, as at the epoch boundaries there is a discontinuity. See also the background on the CTF MEG system.

Magnetic resonance images (MRIs) were obtained with a 1.5 T Siemens system. During the MRI scans ear molds with small containers filled with vitamin E marked the same anatomical landmarks. This allows for realignment of the MRIs to the MEG coordinate system according to the anatomical landmarks.

---

1. Kutas M, Hillyard SA. (1980) Reading senseless sentences: brain potentials reflect semantic incongruity. Science. 207(4427):203-5
2. Kutas M, Federmeier KD. (2000) Electrophysiology reveals semantic memory use in language comprehension. Trends Cogn Sci. 4(12):463-470
3. van den Brink D, Brown CM, & Hagoort P. (2001). Electrophysiological evidence for early contextual influences during spoken-word recognition: N200 versus N400 effects. J Cogn Neurosci. 13(7):967-985
4. Wang L, Jensen O, van den Brink D, Weder N, Schoffelen JM, Magyari L, Hagoort P, Bastiaansen M. (2012) Beta oscillations relate to the N400m during language comprehension. Hum Brain Mapp. 2012 Dec;33(12):2898-912.

Procedure

The following steps are taken in this tutorial:

- Define segments of data of interest (the trial definition) using **ft_definetrial**
- Read the data into MATLAB using **ft_preprocessing**
- Split up the data for the different conditions **ft_selectdata**

Reading and preprocessing the interesting trials

Using the FieldTrip function **ft_definetrial** you can define the pieces of data that will be read in for preprocessing. Trials are defined by their begin and end sample in the data file and each trial has an offset that defines where the relative t=0 point (usually the point of the stimulus-trigger) is for that trial.

The **ft_definetrial** and **ft_preprocessing** functions require the original MEG dataset, which is available at ftp://ftp.fieldtriptoolbox.org/pub/fieldtrip/tutorial/Subject01.zip

Do the trial definition for the all conditions together:

```
cfg                         = [];
cfg.dataset                 = 'Subject01.ds';
cfg.trialfun                = 'ft_trialfun_general'; % this is the default
cfg.trialdef.eventtype      = 'backpanel trigger';
cfg.trialdef.eventvalue     = [3 5 9]; % the values of the stimulus trigger
for the three conditions
% 3 = fully incongruent (FIC), 5 = initially congruent (IC), 9 = fully
congruent (FC)
cfg.trialdef.prestim        = 1; % in seconds
cfg.trialdef.poststim       = 2; % in seconds

cfg = ft_definetrial(cfg);
```

This results in a cfg.trl that contains the trial definitions of all conditions (since we specified all three trigger values: 3, 5, and 9). In cfg.trl the beginning, the trigger offset and the end of each trial relative to the beginning of the raw data are defined. Additionally, cfg.trl contains a column that contains the trigger values, i.e., it tells you to which condition each trial belongs.

The output of **ft_definetrial** can be used for **ft_preprocessing**.

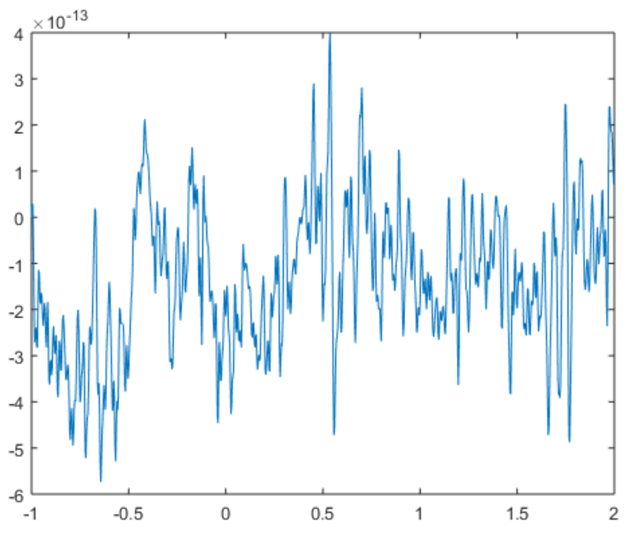
```
cfg.channel    = {'MEG' 'EOG'};
cfg.continuous = 'yes';
data_all = ft_preprocessing(cfg);

Save the data to disk

    save PreprocData data_all
```

The output of **ft_preprocessing** is the structure data_all which has the following fields:

```
data_all =
           hdr: [1x1 struct]
         label: {152x1 cell}
          time: {1x261 cell}
         trial: {1x261 cell}
       fsample: 300
    sampleinfo: [261x2 double]
     trialinfo: [261x1 double]
          grad: [1x1 struct]
           cfg: [1x1 struct]
```

data_all contains a field data_all.trialinfo, which contains the 4th column of the trl (trial definition) which contains the trigger values. The most important fields are data_all.trial containing the individual trials and data_all.time containing the time vector for each trial. To visualize the single trial data (trial 1) on one channel (channel 130) do the following:

```
plot(data_all.time{1}, data_all.trial{1}(130,:))
```



Split up the conditions by selecting trials according to their trigger value (in data_all.trialinfo).

```
cfg=[];
cfg.trials = data_all.trialinfo==3;
dataFIC = ft_selectdata(cfg, data_all);

cfg.trials = data_all.trialinfo==5;
dataIC = ft_selectdata(cfg, data_all);

cfg.trials = data_all.trialinfo==9;
dataFC = ft_selectdata(cfg, data_all);
```

Save the preprocessed data to disk

```
save PreprocData dataFIC dataIC dataFC -append
```

These functions demonstrate how to extract trials from a dataset based on trigger information. Note that some of these trials will be contaminated with various artifact such as eye blinks or MEG sensor jumps. Artifact rejection is described in Preprocessing - Visual artifact rejection

Use your own function for trial selection

There are often cases in which it is not sufficient to define a trial only according to a given trigger signal. For instance you might want to accept or reject a trial according to a button response recorded by the trigger channel as well. Another example might be that you want the signals from the EMG or A/D channel being part of the trial selection. In those cases it is necessary to define a specialized function for trial selections. In this example we only select trials of which the previous trial was of a different condition. First, the condition of the current and the preceding trial are noted in the trial information. At the end, all trials in which these are the same, are removed. This is helpful when for example, you are interested in sequential trial effects, like: Is the signal different when it was preceded by a trial of type A rather than a trial of type B?

```
function trl = mytrialfun(cfg);

% this function requires the following fields to be specified
% cfg.dataset
% cfg.trialdef.eventtype
% cfg.trialdef.eventvalue
% cfg.trialdef.prestim
% cfg.trialdef.poststim

hdr   = ft_read_header(cfg.dataset);
event = ft_read_event(cfg.dataset);

trl = [];

for i=1:length(event)
if strcmp(event(i).type, cfg.trialdef.eventtype)
  % it is a trigger, see whether it has the right value
  if ismember(event(i).value, cfg.trialdef.eventvalue)
    % add this to the trl definition
    begsample     = event(i).sample - cfg.trialdef.prestim*hdr.Fs;
    endsample     = event(i).sample + cfg.trialdef.poststim*hdr.Fs - 1;
    offset        = -cfg.trialdef.prestim*hdr.Fs;
    trigger       = event(i).value; % remember the trigger (=condition) for
each trial
    if isempty(trl)
      prevtrigger = nan;
    else
      prevtrigger   = trl(end, 4); % the condition of the previous trial
    end
    trl(end+1, :) = [round([begsample endsample offset])  trigger
prevtrigger];
  end
end
end

samecondition = trl(:,4)==trl(:,5); % find out which trials were preceded by
a trial of the same condition
trl(samecondition,:) = []; % delete those trials
```

Save the trial function together with your other scripts as mytrialfun.m. To ensure that **ft_preprocessing** is making use of the new trial function use the commands

```
cfg = [];
cfg.dataset                = 'Subject01.ds';
cfg.trialfun               = 'mytrialfun';     % it will call your function and
pass the cfg
cfg.trialdef.eventtype  = 'backpanel trigger';
cfg.trialdef.eventvalue = [3 5 9];            % read all conditions at once
cfg.trialdef.prestim    = 1; % in seconds
cfg.trialdef.poststim   = 2; % in seconds

cfg = ft_definetrial(cfg);

cfg.channel = {'MEG' 'STIM'};
dataMytrialfun = ft_preprocessing(cfg);
```

When you do not specify cfg.trialfun, **ft_definetrial** will call a function named trialfun_general as default. Then trials will be defined as we have seen it in the earlier section (Reading and preprocessing the interesting trials).

The output dataMytrialfun now contains fewer trials than before: 192 instead of 261. Thus, we discarded 69 trials that had the same condition in the previous trial. The field dataMytrialfun.trialinfo contains the 4th column of the trl (trial definition) (trigger values of the current trial), and the 5th column of the trl (trigger values of the previous trial).

```
dataMytrialfun =

          hdr: [1x1 struct]
        label: {152x1 cell}
         time: {1x192 cell}
        trial: {1x192 cell}
      fsample: 300
   sampleinfo: [192x2 double]
    trialinfo: [192x2 double]
         grad: [1x1 struct]
          cfg: [1x1 struct]
```

More on the trialinfo field can be found in the faq.

Suggested further reading

After having finished this tutorial on preprocessing, you can continue with the event related averaging or with the time-frequency analysis tutorial.

If you have more questions about preprocessing, you can also read the following FAQs:

- What dataformats are supported?
- How can I deal with a discontinuous Neuralynx recording?
- How can I consistently represent artifacts in my data?
- How can I convert one dataformat into an other?
- How can I extend the reading functions with a new dataformat?
- How can I find out what eventvalues and eventtypes there are in my data?
- How can I import my own data format?
- How can I interpret the different types of padding in FieldTrip?
- How can I merge two datasets that were acquired simultaneously with different amplifiers?
- How can I preprocess a dataset that is too large to fit into memory?
- How can I process continuous data without triggers?
- How can I read all channels from an EDF file that contains multiple sampling rates?
- How can I read corrupted (unsaved) CTF data?
- How does the filter padding in preprocessing work?
- I have problems reading in neuroscan .cnt files. How can I fix this?
- Reading is slow, can I write my raw data to a more efficient file format?
- How can I check or decipher the sequence of triggers in my data?
- What is the relation between "events" (such as triggers) and "trials"?
- What kind of filters can I apply to my data?
- Why am I not getting integer frequencies?
- Why does my TFR look strange (part I, demeaning)?
- Why does my TFR look strange (part II, detrending)?
- Why is there a residual 50Hz line-noise component after applying a DFT filter?

and example scripts:

- Detect the muscle activity in an EMG channel and use that as trial definition
- Determine the filter characteristics
- Fixing a missing channel
- Getting started with reading raw EEG or MEG data
- Making your own trialfun for conditional trial definition
- Re-reference EEG and iEEG data
- Use denoising source separation (DSS) to remove ECG artifacts
- Use independent component analysis (ICA) to remove ECG artifacts
- Use independent component analysis (ICA) to remove EOG artifacts