

# Getting started with EEG data, quality checks and ERPs

## Introduction

In FieldTrip the preprocessing of data refers to the reading of the data, segmenting the data around interesting events such as triggers, temporal filtering, artifact rejection 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, only read those segments from the data file and apply the filters each segment. The remainder of this tutorial explains the second approach; it is in general more memory efficient also works for large data sets.

Please note that the dataset used in this tutorial is not particularly large; we could also have used the first approach. But we want to demonstrate how to work in general with large data (or with computers that have a small memory). You can see the first approach used in the tutorial on [reading continuous EEG](#).

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 or according to your own criteria when you write your own trial function.

## Procedure

In this tutorial the following steps will be taken:

- Read the data into MATLAB using [ft\\_definetrial](#) and [ft\\_preprocessing](#)
- Extract bipolar EOG channels with [ft\\_preprocessing](#) and get rid of reference channels with [ft\\_selectdata](#). Combine EOG and data channels with [ft\\_appenddata](#).
- Visual artifact rejection with [ft\\_databrowser](#) and [ft\\_rejectvisual](#).
- Computing trial averages with [ft\\_timelockanalysis](#).
- Plotting ERPs with [ft\\_topoplotER](#)

## Preprocessing

You can download the single-subject task dataset [here](#) from our FTP server.

### Reading trials

Let us first look at the different trigger codes present in the dataset

```
cfg = [];  
cfg.dataset =  
'madrid2019/tutorial_erp/single_subject_task/raw/subj2.vhdr';  
cfg.trialdef.eventtype = '?';  
dummy = ft_definetrial(cfg);
```

This will display the event types and values on screen. The trigger codes S112, S122, S132, S142 (animals) S152, S162, S172, S182 (tools) correspond to the presented visual stimuli. The trigger codes S113, S123, S133, S143 (animals) S153, S163, S173, S183 (tools) correspond to the presented auditory stimuli. These are the triggers we will select for now.

- [Introduction](#)
- [Procedure](#)
- [Preprocessing](#)
  - [Reading trials](#)
  - [Extracting EOG channels](#)
- [Visual artifacts detection](#)
  - [Display one trial at a time](#)
  - [Display one channel at a time](#)
  - [Display a summary](#).
  - [Inspect cleaned data](#)
- [Computing and plotting the ERPs](#)
  - [Channel layout](#)
  - [Trial-average](#)
- [Next steps](#)
- [See also](#)

```

trigVIS = {'S112', 'S122', 'S132', 'S142', 'S152', 'S162', 'S172', 'S182'};
trigAUD = {'S113', 'S123', 'S133', 'S143', 'S153', 'S163', 'S173', 'S183'};

cfg = [];
cfg.dataset = 'single_subject_task/raw/subj2.vhdr';
cfg.trialdef.eventtype = 'Stimulus';
cfg.trialdef.eventvalue = [trigVIS trigAUD];
cfg.trialdef.prestim = 0.5;
cfg.trialdef.poststim = 0.8;
cfg = ft_definetrial(cfg);

```

After the call to **ft\_definetrial**, the configuration structure `cfg` now not only stores the dataset name, but also contains a field `cfg.tr1` with the definition of the segments of data that will be used for further processing and analysis. Each row in the `tr1` matrix represents a single epoch-of-interest, and the `tr1` matrix has at least 3 columns. The first column defines (in samples) the begin sample of each epoch with respect to how the data are stored in the raw datafile. The second column defines the end sample of each epoch, and the third column specifies the offset of the first sample within each epoch with respect to time 0 within that epoch.

We will now read in the trials that were defined in the `cfg` structure. For preprocessing this EEG data set, the choice of the reference has to be considered. During acquisition the reference channel of the EEG amplifier was attached to the left mastoid. We would like to analyze this data with a linked-mastoid reference (also known as an average-mastoid reference).

```

cfg.reref = 'yes';
cfg.channel = 'all';
cfg.implicitref = 'M1'; % the implicit (non-recorded) reference channel is
                        added to the data
cfg.refchannel = {'M1', '53'}; % the average of these will be the new reference,
                               note that '53' corresponds to the right mastoid (M2)

```

#### Filtering options

```

cfg.lpfilter = 'yes';
cfg.lpfreq = 100;
cfg.demean = 'yes';
cfg.baselwindow = [-inf 0];
data = ft_preprocessing(cfg);

```

For consistency we will rename the channel with the name 53 located at the right mastoid to M2

```

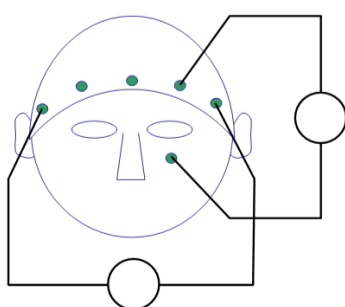
chanindx = find(strcmp(data.label, '53'));
data.label{chanindx} = 'M2';

```

#### Extracting EOG channels

We now continue with re-referencing to extract the bipolar EOG signal from the data. For the vertical EOG we will use channel 50 and channel 64. For the horizontal EOG we will compute the difference between the potential recorded in channels 51 and 60.

Some acquisition systems, such as the CTF system or the Biosemi system, allow for direct bipolar recording of ECG, EOG and EMG. The re-referencing step to obtain the EOG is therefore not required when working with Biosemi or other bipolar data.



Select the channels that we need for the vertical EOG

- [Introduction](#)
- [Procedure](#)
- [Preprocessing](#)
  - [Reading trials](#)
  - [Extracting EOG channels](#)
- [Visual artifacts detection](#)
  - [Display one trial at a time](#)
  - [Display one channel at a time](#)
  - [Display a summary.](#)
  - [Inspect cleaned data](#)
- [Computing and plotting the ERPs](#)
  - [Channel layout](#)
  - [Trial-average](#)
- [Next steps](#)
- [See also](#)

```

cfg          = [];
cfg.channel  = {'50' '64'};
cfg.reref    = 'yes';
cfg.implicitref = []; % this is the default, we mention it here to be explicit
cfg.refchannel = {'50'};
eogv         = ft_preprocessing(cfg, data);

```

only keep one channel, and rename it to EOGV

```

cfg          = [];
cfg.channel  = '64';
eogv         = ft_selectdata(cfg, eogv);
eogv.label   = {'eogv'};

```

Select the channels that we need for the horizontal EOG

```

cfg          = [];
cfg.channel  = {'51' '60'};
cfg.reref    = 'yes';
cfg.implicitref = [];
cfg.refchannel = {'51'};
eogh         = ft_preprocessing(cfg, data);

```

only keep one channel, and rename to it eogh

```

cfg          = [];
cfg.channel  = '60';
eogh         = ft_selectdata(cfg, eogh);
eogh.label   = {'eogh'};

```

We can now discard the extra channels from the original recording that were used to pick up the EOG and add the bipolar-referenced EOGv and EOGh channels that we have just created.

Keep only the EEG channels

```

cfg          = [];
cfg.channel = setdiff(1:60, [50, 51, 60, 64]); % you can use either strings or
numbers as selection
data         = ft_selectdata(cfg, data);

```

Append the EOGH and EOGV channel to the selected EEG channels

```

cfg = [];
data = ft_appenddata(cfg, data, eogv, eogh);

```

FieldTrip data structures are intended to be "lightweight", in the sense that the internal MATLAB arrays can be transparently accessed. Have a look at the data as you read it into memory

```

>> data

data =

    hdr: [1x1 struct]
   label: {1x65 cell}
    time: {1x400 cell}
   trial: {1x400 cell}
 fsample: 500
sampleinfo: [400x2 double]
  trialinfo: [400x1 double]
    cfg: [1x1 struct]

```

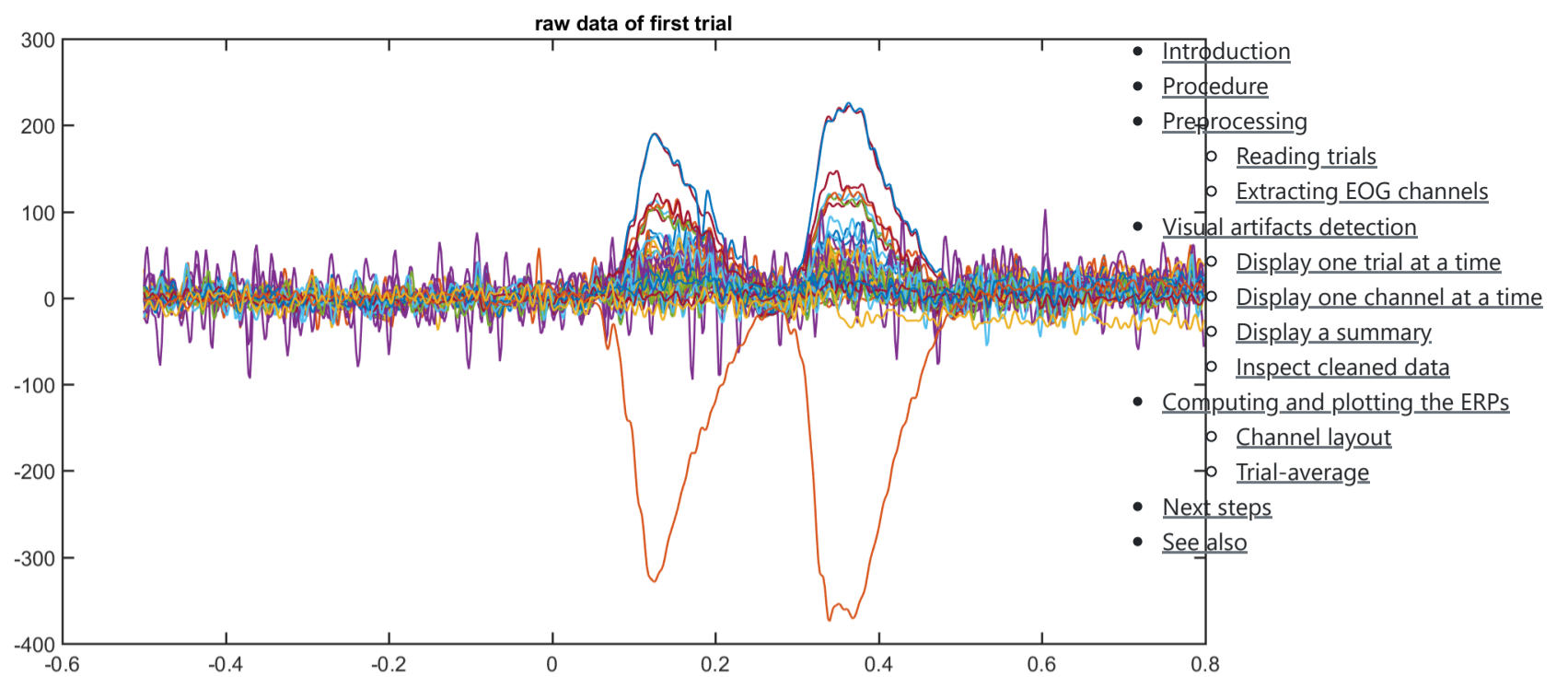
The fields in the raw data structure are explained in more detail in [ft\\_datatype\\_raw](#). You can use the default MATLAB plot functions to show the EEG data for all channels in for example the first trial:

```

plot(data.time{1}, data.trial{1});

```

- [Introduction](#)
- [Procedure](#)
- [Preprocessing](#)
  - [Reading trials](#)
  - [Extracting EOG channels](#)
- [Visual artifacts detection](#)
  - [Display one trial at a time](#)
  - [Display one channel at a time](#)
  - [Display a summary.](#)
  - [Inspect cleaned data](#)
- [Computing and plotting the ERPs](#)
  - [Channel layout](#)
  - [Trial-average](#)
- [Next steps](#)
- [See also](#)



However, a more efficient way to quickly visualize and scroll through your data is to use [ft\\_databrowser](#). That function also allows you to mark the time windows in which artifacts are present.

## Visual artifacts detection

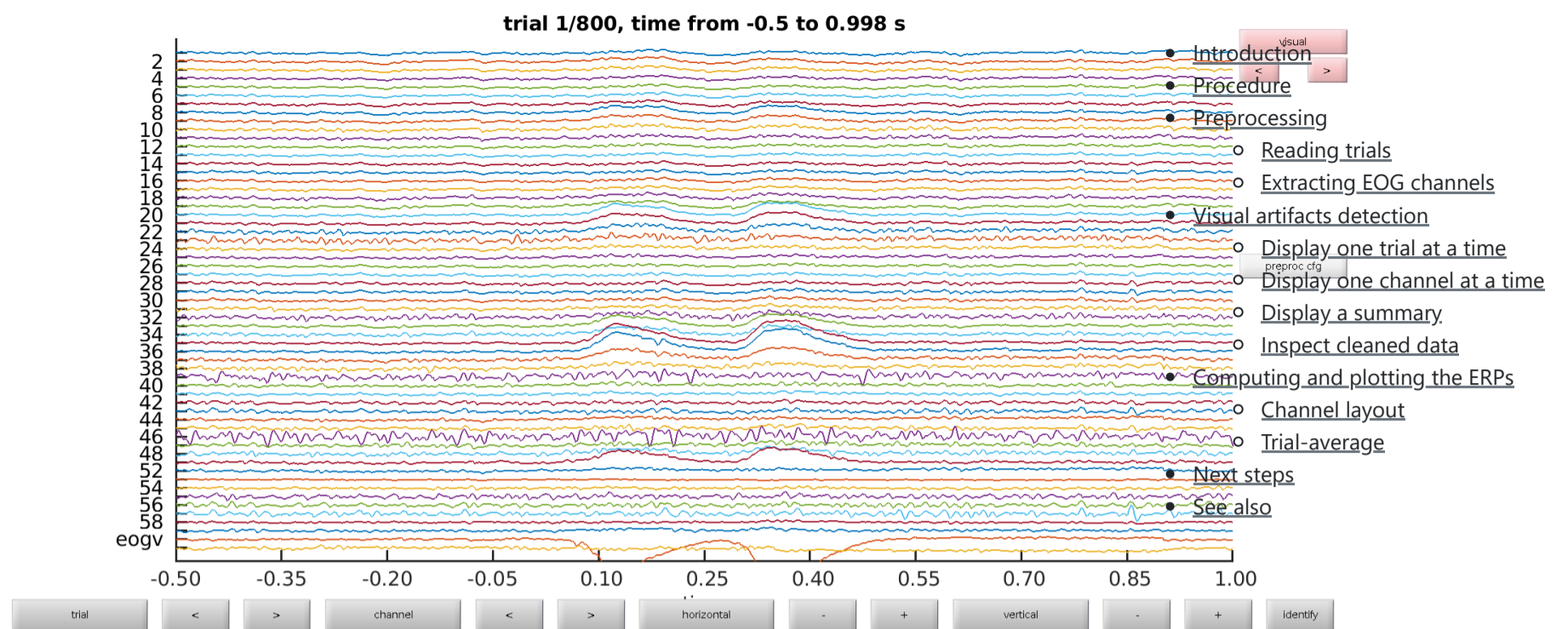
When using visual inspection to detect and reject artifacts, keep in mind that it is often a subjective decision to reject certain trials and to keep others. The type of artifacts to reject depends on the follow-up analysis you would like to do. For frequency analysis of power in the gamma band it is important to reject all trials with (high frequency) muscle artifacts, but for a ERF analysis it is more important to reject trials with (low frequency) drifts and eye artifacts.

It is important that you make your selection of trials *blind* to the experimental manipulation. E.g. in an oddball experiment when you have many standards and few oddballs, you might be tempted to be more strict on the standards and more lenient on the oddballs. This would result in more remaining artifacts in your oddball condition, which could cause a trivial difference in the ERP waveforms.

So you should apply the same criteria for artifact rejection to all your experimental conditions.

To get a first impression of our data quality we will use [ft\\_databrowser](#) to look at our individual trials. This is a good option if you want to quickly browse your data, as it takes both continuous and segmented data as inputs. With [ft\\_databrowser](#) you can display all channels at the same time to inspect non-systematic artifacts such as blinks or electrode movement.

```
cfg = [];
cfg.viewmode = 'vertical';
artfct      = ft_databrowser(cfg, data)
```



### Exercise 1

Skip through a couple of data segments and see if you can already spot some artifacts. Use the buttons to mark an artifact. Are there any bad channels in this dataset?

As you might have noticed it is easy to mark segments with [ft\\_databrowser](#), but it is not possible to mark channels as bad; also looking at each trial individually and then clicking with the mouse to go to the next is not always very efficient.

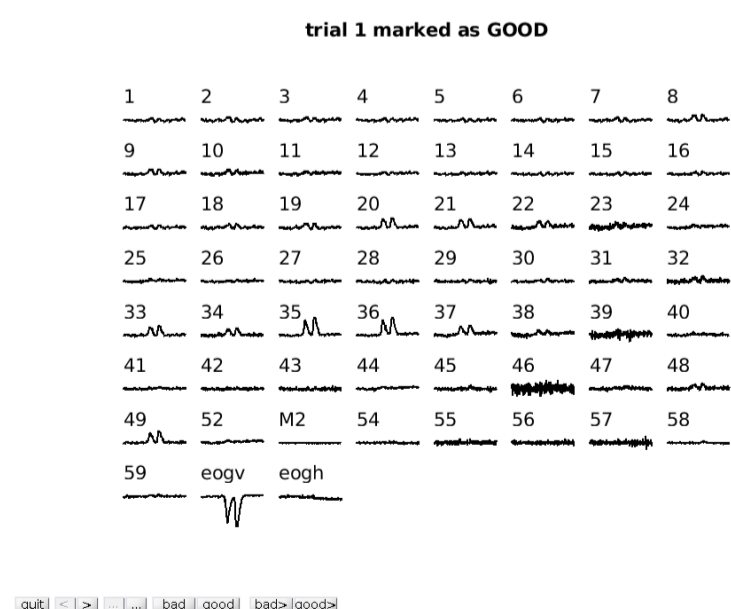
### Display one trial at a time

Another option for visual artifact detection is [ft\\_rejectvisual](#). This function allows you to inspect data that has been segmented into trials and identify either bad trials or bad channels. It allows you to browse through large amounts of data in a single figure by either showing all channels at once (per trial), or showing all trials at once (per channel), or by showing a summary of all channels and trials.

We will first call [ft\\_rejectvisual](#) with all channels at once. This corresponds more or less to the way that [ft\\_databrowser](#) displays the data (one trial at the time), but now we can see individual channels better and can mark channels that are bad.

```
cfg = [];
cfg.method = 'trial';
data_clean = ft_rejectvisual(cfg, data);
```

To mark a channel that is bad, you have to click in the figure on that specific channel. Clicking the channel once more toggles it back to good. To mark the whole trial as bad, you have to click on the "bad" button at the bottom of the figure. Click through the trials using the > button to inspect each trial.



### Exercise 2

Can you spot which channels are noisier than others? Using the mouse, you can select channels that you want to remove from the data.

The **ft\_rejectvisual** function directly returns the clean data with the bad channels and trials removed and you do not have to call **ft\_rejectartifact** or **ft\_rejectcomponent**. We could continue working with the cleaned data, but for now we will only save the list of channel names of the the bad channels and continue with our data inspection.

```
bad_chan = setdiff(data.label, data_clean.label);
```

### Display one channel at a time

Next we will use **ft\_rejectvisual** to display all trials for a single channel; this is helpful to identify whether the channel is noisy in general, or becomes bad at a certain time during the experiment, or only has some glitches. It is also helpful to inspect the EOG channels to identify trials that have eye blinks or eye movements.

```
cfg = [];
cfg.method = 'channel';
cfg.channel = {bad_chan{:}, 'eogv', 'eogh'};
data_clean = ft_rejectvisual(cfg, data);
```

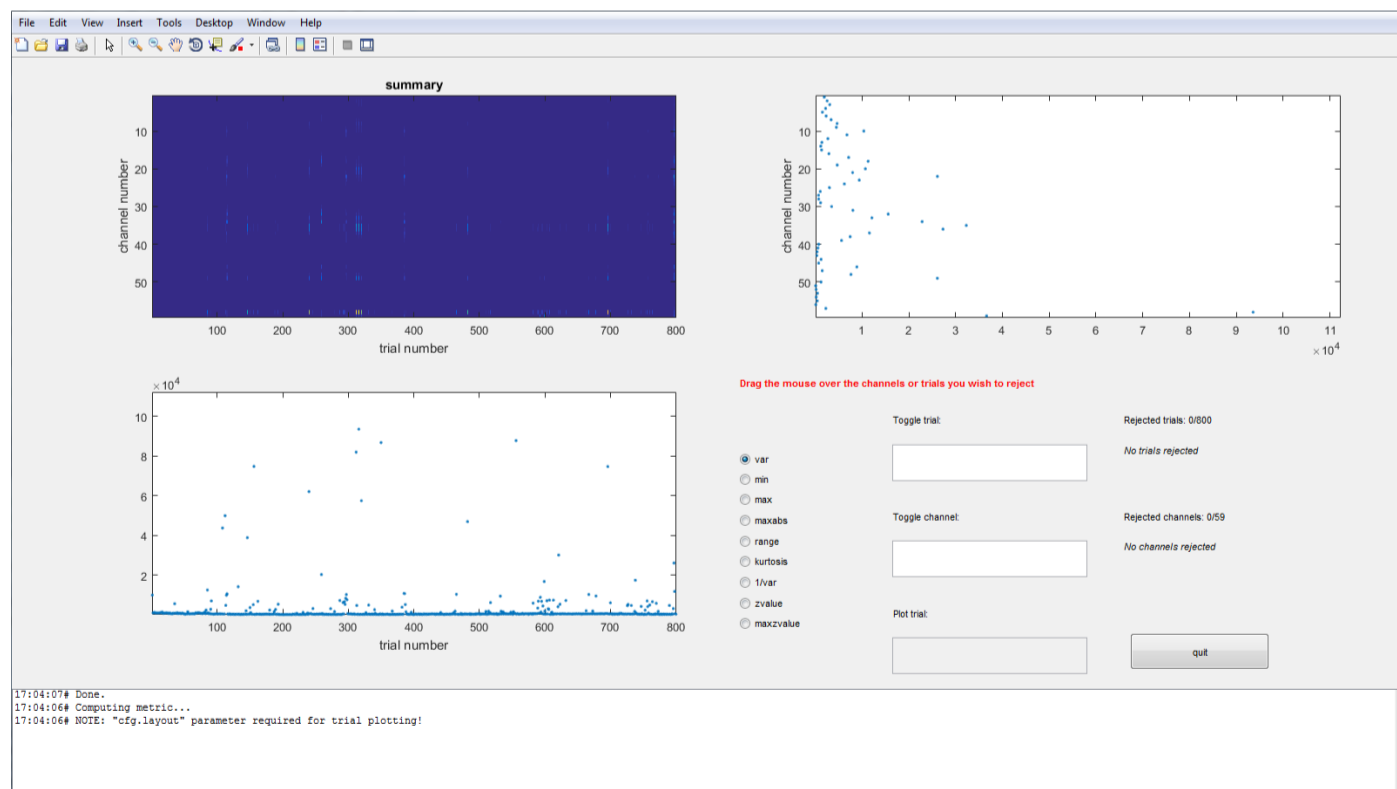
Since we removed some bad trials, the cleaned data will have fewer trials than the original. Let us identify and store the trial numbers that have blinks. We are using the `trialinfo` field for this, which contains the begin and end sample relative to the original data file.

```
bad_trial = ismember(data.sampleinfo(:,1), setdiff(data.sampleinfo(:,1),
data_clean.sampleinfo(:,1)));
```

### Display a summary

Finally we will call **ft\_rejectvisual** one more time in 'summary' mode. This option is the fastest to identify either trials or channels that are noisy.

```
cfg = [];
cfg.method = 'summary';
cfg.channel = 'all';
cfg.trials = ~bad_trial; % exclude the trials that we already identified as bad
data_clean = ft_rejectvisual(cfg, data);
```



### Exercise 3

Which channels show the most variance? Why is that?

If you would like to keep track of which trials you reject, keep in mind that the trial numbers change when you call **ft\_rejectvisual** more than once, or if you use the `cfg.trials` option for selection. If you want to keep track which data you rejected, it is best look at the sample numbers in the `trialinfo` field. These sample numbers always refer to the original EEG data file.

- [Introduction](#)
- [Procedure](#)
- [Preprocessing](#)
  - [Reading trials](#)
  - [Extracting EOG channels](#)
- [Visual artifacts detection](#)
  - [Display one trial at a time](#)
  - [Display one channel at a time](#)
  - [Display a summary.](#)
  - [Inspect cleaned data](#)
- [Computing and plotting the ERPs](#)
  - [Channel layout](#)
  - [Trial-average](#)
- [Next steps](#)
- [See also](#)

## Inspect cleaned data

Now that we have identified artifacts using different visual inspection approaches, we will use **ft\_databrowser** again to inspect the data with bad trials marked as such. For this we first compare the `sampleinfo` between the original and the clean data structure.

```
bad_samples = data.sampleinfo(ismember(data.sampleinfo(:,1),
setdiff(data.sampleinfo(:,1), data_clean.sampleinfo(:,1))),:);

cfg = [];
cfg.viewmode = 'vertical';
cfg.artfctdef.visual.artifact = bad_samples;
cfg_artif = ft_databrowser(cfg, data);
```

If the cleaning with **ft\_rejectvisual** was not sufficient yet, you can continue to mark trials or data segments as artifacts within **ft\_databrowser**. The updated artifact definition will be saved in the `cfg_artif` output and you can subsequently reject those artifacts using **ft\_rejectartifact**. You can see an example of this in the [resting-state cleaning EEG tutorial](#)

## Computing and plotting the ERPs

### Channel layout

For topographic plotting and sometimes for analysis it is necessary to know precisely where the electrodes were positioned on the scalp. In contrast to the sensor arrangement in the helmet from a given MEG manufacturer, the topographical arrangement of the channels in an EEG system is not fixed. You can use the acquisition system with different caps, and the number and position of electrodes can be adjusted depending on the experimental question. In the current data, a cap with a so-called equidistant electrode layout was used for a 64-channel active electrode system (BrainVision ActiCap).

The channel positions are in general not recorded during EEG measurements (although, see [this](#) tutorial) and are hence not stored in the EEG dataset. For plotting you have to use a layout file; this is a `.mat` file that contains the 2-D positions of the channels. FieldTrip provides a number of template layouts for different EEG caps in the `fieldtrip/template/layout` directory. It is also possible to create custom layouts for your own EEG cap, see [ft\\_prepare\\_layout](#) and the [layout tutorial](#).

```
cfg = [];
cfg.layout = 'easycapM10.mat';
ft_layoutplot(cfg)
```

### Trial-average

We will now compute the ERPs for two conditions: visual and auditory stimulus presentation.

For each trial, the condition information is kept with the data structure in `data.trialinfo`. We use **ft\_timelockanalysis** to compute the ERPs and average only those trials in which the trigger code corresponds to the condition of interest (i.e. visual or auditory).

```
cfg = [];
cfg.trials = ismember(data_clean.trialinfo, [112, 122, 132, 142]);
timelockVIS = ft_timelockanalysis(cfg, data_clean);

cfg = [];
cfg.trials = ismember(data_clean.trialinfo, [113, 123, 133, 143]);
timelockAUD = ft_timelockanalysis(cfg, data_clean);
```

### Exercise 4

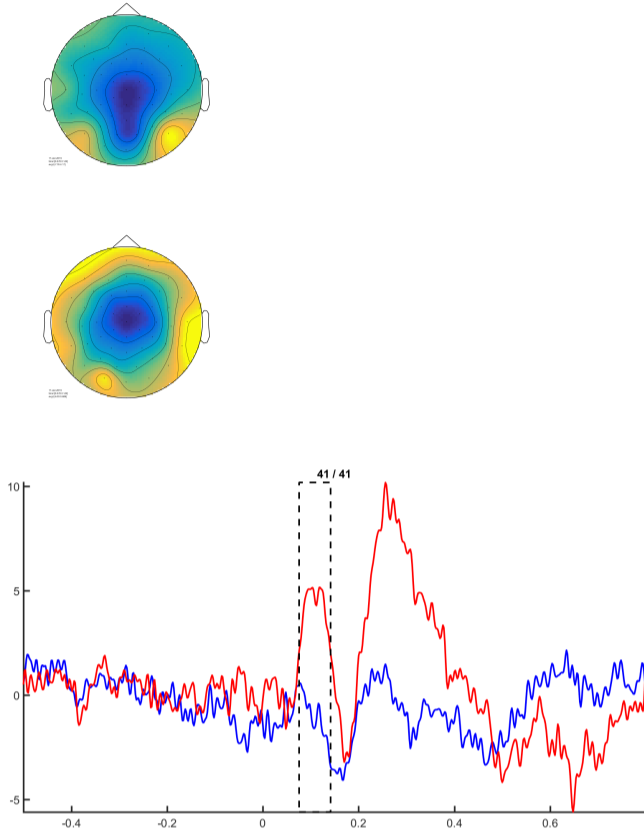
Inspect the resulting data structure after **ft\_timelockanalysis**.

Which fields have been added and what information do they contain? The fields in the data structure are explained in more detail in [ft\\_datatype\\_timelock](#).

- [Introduction](#)
- [Procedure](#)
- [Preprocessing](#)
  - [Reading trials](#)
  - [Extracting EOG channels](#)
- [Visual artifacts detection](#)
  - [Display one trial at a time](#)
  - [Display one channel at a time](#)
  - [Display a summary](#)
  - [Inspect cleaned data](#)
- [Computing and plotting the ERPs](#)
  - [Channel layout](#)
  - [Trial-average](#)
- [Next steps](#)
- [See also](#)

We will now use **ft\_topoplotER** to plot both conditions. By setting the option `cfg.interactive='yes'` you can interact with the figure by selecting certain channels or time windows and going back and forth between the topographic and time representation of the data.

```
cfg = [];
cfg.layout = 'easycapM10.mat';
cfg.interactive = 'yes';
cfg.baseline = [-0.2 0];
ft_topoplotER(cfg, timelockAUD, timelockVIS)
```



### Exercise 5

Select the time window where the conditions differ the most. Do the topographies look as you would expect?

## Next steps

Following the computation of ERPs, FieldTrip offers many more functions to continue analyzing your data. You can use **ft\_math** to compute difference waves, **ft\_timelockstatistics** to run statistics on your ERP effect, compute group level averages with **ft\_timelockgrandaverage** or explore different ways of visualizing, i.e. **ft\_multiplotER** etc.

### Exercise 6

The following code allows you to compute at the ERP difference waves.

```
cfg = [];
cfg.operation = 'subtract';
cfg.parameter = 'avg';
difference = ft_math(cfg, timelockAUD, timelockVIS);
```

This in principle does the same as

```
difference = timelockAUD; % copy one of the
structures
difference.avg = timelockAUD.avg - timelockVIS.avg; % compute the difference
ERP
```

but **ft\_math** will keep provenance (i.e. historical information) in the `cfg` structure, whereas if you do it by hand you will lose the information, which might make it more difficult later in your analysis to figure out what you exactly did to your data.

- [Introduction](#)
- [Procedure](#)
- [Preprocessing](#)
  - [Reading trials](#)
  - [Extracting EOG channels](#)
- [Visual artifacts detection](#)
  - [Display one trial at a time](#)
  - [Display one channel at a time](#)
  - [Display a summary.](#)
  - [Inspect cleaned data](#)
- [Computing and plotting the ERPs](#)
  - [Channel layout](#)
  - [Trial-average](#)
- [Next steps](#)
- [See also](#)

## See also



## Frequently asked questions:

- [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?](#)
  
- [What kind of filters can I apply to my data?](#)

## Examples:

- [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](#)
- [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](#)
  
- [Fit a dipole to the tactile ERF after mechanical stimulation](#)

- [Introduction](#)
- [Procedure](#)
- [Preprocessing](#)
  - [Reading trials](#)
  - [Extracting EOG channels](#)
- [Visual artifacts detection](#)
  - [Display one trial at a time](#)
  - [Display one channel at a time](#)
  - [Display a summary.](#)
  - [Inspect cleaned data](#)
- [Computing and plotting the ERPs](#)
  - [Channel layout](#)
  - [Trial-average](#)
- [Next steps](#)
- [See also](#)

Edit [this page on GitHub](#).

This page last modified: 2020-05-28 16:41 +0200

Copyright (C) 1999-2020; [© License](#); [Privacy](#).