

FIELDTRIP TOOLBOX TUTORIAL REPORT

PRADEEP KUMAR YADAV | BM18MTECH11005

TUTORIAL 2-

Dealing with artifacts

Background: what is an artifact?

- Some unwanted and unexpected feature in data.
- Artifacts are undesired data which is acquired from EEG and MEG.
- It may be physiological or non_physiological from origin.

How does FieldTrip manage artifacts?

first identify the artifacts and then subsequently removing them.

Artifacts can be visualize or can be predicted when we see the response .

- rejecting the piece of data containing the artifact (e.g. for a short-lived artifact)
- subtracting the **spatio-temporal contribution** of the artifact from the data (e.g. for line noise)

fieldtrip provide the the function to detect the artifacts whether data is continuous or trial-based depending on whether your data is stored on **disk or already in memory**.

Detecting the artifacts in continuous data to apply filters(band pass filter) and zoom out the recording.

Rejecting segments of data containing artifacts

In this type of artifact detection and rejection, pieces of data that contain artifacts are identified and removed from the data set

Manual/visual detection

In this manually and visualize the artifacts in continuous data trails.

And identify the affected with artifacts and reject the artifact.

The functions that are available for manual artifact detection are

ft rejectvisual

The **ft_rejectvisual** function works only for segmented data (i.e. trials) that have already been read into memory.

FT_REJECTVISUAL shows the preprocessed data in **all** channels **and/or** trials to allow the user to make a visual selection of the data that should be rejected. The data can be displayed in a **"summary"** mode, in **which case** the variance (**or** another metric) in each channel **and** each trial **is** computed. Alternatively, **all** channels can be shown at once allowing paging through the trials, **or all** trials can be shown, allowing paging through the channels.

ft databrowser

The **ft_databrowser** function works both for continuous and segmented data and also works with the data either on disk or already read into memory.

FT_DATABROWSER can be used **for** visual inspection of data. Artifacts that were detected by artifact **functions** (see FT_ARTIFACT_xxx **functions** where xxx **is** the **type** of artifact) are marked. Additionally data pieces can be marked **and** unmarked as artifact by manual selection. The output **cfg** **contains** the updated specification of the artifacts.

After detecting the time-segments with the artifacts, you should call **ft_rejectartifact** to remove them from your data (when the data is already in memory) or from your trial definition (when the data is still on disk).

a proper ICA unmixing of your data requires that the atypical artifacts (e.g. electrode movement, squid jumps) are removed **prior** to calling **ft_componentanalysis**. After you have determined what the bad components are, you can call **ft_rejectcomponent** to project the data back to the sensor level, excluding the bad components.

Automatic detection

Fieldtrip contains collection of functions to detect the artifacts automatically.

Although the automatic artifact detection algorithm works efficiently for well-known artifacts in well-behaved data, you should **not** use the automatic detection functions as your **default** method.

The available functions for automatic artifact detection are:

- [ft_artifact_clip](#)
- [ft_artifact_ecg](#)
- [ft_artifact_threshold](#)
- [ft_artifact_eog](#)
- [ft_artifact_jump](#)
- [ft_artifact_muscle](#)
- [ft_artifact_zvalue](#)

eog, jump and muscle detection functions are all just wrappers around [ft_artifact_zvalue](#) where the filter and **padding options are set to reasonable defaults.**

Removing artifacts from the data

If you use manual or automatic detection of time segments that contain an artifact, you usually would proceed to reject those segments from subsequent analysis with [ft_rejectartifact](#)

Subtracting spatial/temporal/spectral aspects of data reflecting artifacts

In this type of artifact detection and rejection, spatial/temporal/spectral aspects of the data that contain artifacts are identified and removed from the data set. For example, **certain spectral components such as line noise,** are subtracted from the data.

Using ICA-

Another commonly used approach is to make a linear decomposition of the data using methods such as ICA (independent component analysis) or PCA (principal component analysis). These methods consist of applying a set of **spatial filters to the data,** after which the data is no longer represented at the level of recorded (scalp) channels, but as a set of virtual channels or components.

If you use ICA to detect artifacts, you usually would proceed with projecting the decomposed data (excluding the artifact components) back to the sensor level. This is done with **`ft_rejectcomponent`**.

Further I will discuss more about ICA and how to detect and remove the artifacts.

Visual artifact rejection

Introduction

First we have to load the preprocessing data in matlab

```
load PreprocData dataFIC
```

for accurate measurement we should have clean or denoised data.

One of the factors that is difficult to control are the presence of artifacts in the data. These artifact are physiological or can result from the acquisition electronics. The strongest physiological artifacts stem from eye blinks, eye movements and head movements. Muscle artifact from swallowing and neck contraction can be a problem as well. Artifacts related to the electronics are 'SQUID jumps' or spikes seen in several channels. To start with, it is best to avoid those artifacts during the recording. You can instruct the subject not to blink during the trial, but instead give him some well-defined time between the trials in which he is allowed to blink. But of course there will always be some artifacts in the raw data.

keep in mind that it is a subjective decision to reject certain trials and keep other trials. Some data are bad and some data are good.

time-frequency analysis of power in the gamma band it is important to reject all trials with muscle artifacts, but for a ERF analysis it is more important to reject trials with drifts and eye artifacts.

The functions that are available for visual artifact detection are

- **`ft_rejectvisual`**
- **`ft_databrowser`**

already I have discussed above.

Procedure-

The following steps are taken to do visual artifact rejection

- Read the data into MATLAB using [ft_definetrial](#) and [ft_preprocessing](#), as explained in the [previous tutorial](#)
- Visual inspection of the trials and rejection of artifacts using [ft_rejectvisual](#)
- Alternatively: use [ft_databrowser](#) and mark the artifacts manually by interactively paging trial by trial

Manual artifact rejection - display one trial at a time

The configuration option `cfg.method` provides the possibility of browsing through the data channel by channel (`cfg.method = 'channel'`), trial by trial (`cfg.method = 'trial'`).

The scaling of the plots is automatically adjusted according to the maximum amplitude over all channels. The scaling can be set using `cfg.alim`. For EOG/EEG channels `cfg.alim=5e-5` (50 micro Volt) is a useful scale and for the MEG channels `cfg.alim=1e-12` (10 fT/cm).

To browse through the data trial by trial while viewing all channels write

This is for MEG channels

```
cfg      = [];  
cfg.method = 'trial';  
cfg.alim  = 1e-12;  
dummy    = ft_rejectvisual(cfg,dataFIC);
```

the input is raw data with 152 channels and 87 trials

the call to "ft_selectdata" took 1 seconds

showing the data per trial, all channels at once

trial 1 marked as GOOD

trial 1 marked as BAD

trial 1 marked as GOOD

trial 1 marked as BAD

trial 2 marked as GOOD

trial 2 marked as GOOD

trial 3 marked as GOOD

trial 3 marked as BAD

trial 3 marked as GOOD

trial 4 marked as GOOD

trial 4 marked as GOOD

trial 5 marked as GOOD

trial 5 marked as GOOD

trial 6 marked as GOOD

trial 6 marked as GOOD

trial 7 marked as GOOD

trial 7 marked as GOOD

trial 8 marked as GOOD

trial 8 marked as GOOD

trial 9 marked as GOOD

trial 9 marked as GOOD

trial 10 marked as GOOD

trial 10 marked as GOOD

trial 11 marked as GOOD

channel MLC11 marked as BAD

trial 11 marked as GOOD

channel MLC11 marked as GOOD

trial 11 marked as GOOD

channel MLC11 marked as BAD

trial 11 marked as GOOD

channel MLC11 marked as GOOD

trial 11 marked as GOOD

channel MLC11 marked as BAD

channel MLC11 marked as GOOD

channel MLC11 marked as BAD

channel MLC11 marked as GOOD

trial 11 marked as GOOD

channel MLC11 marked as BAD

channel MLF44 marked as BAD

trial 11 marked as GOOD

channel MLF44 marked as GOOD

channel MLF44 marked as BAD

trial 11 marked as GOOD

channel MLF44 marked as GOOD

trial 11 marked as GOOD

trial 12 marked as GOOD

trial 12 marked as GOOD

trial 12 marked as BAD

trial 12 marked as GOOD

trial 13 marked as GOOD

trial 13 marked as GOOD

trial 13 marked as GOOD

channel MLC12 marked as BAD

trial 13 marked as GOOD

channel MLC12 marked as GOOD

channel MLC12 marked as BAD

trial 13 marked as GOOD

channel MLC12 marked as GOOD

trial 13 marked as GOOD

channel MLC12 marked as BAD

channel MLC12 marked as GOOD

trial 13 marked as GOOD

channel MLC12 marked as BAD

channel MLC12 marked as GOOD

trial 13 marked as GOOD

channel MLC12 marked as BAD

channel MLC12 marked as GOOD

channel MLC12 marked as BAD

trial 13 marked as GOOD

channel MLC12 marked as GOOD

channel MLC12 marked as BAD

trial 13 marked as GOOD

channel MLC12 marked as GOOD

unknown key pressed

unknown key pressed

unknown key pressed

trial 13 marked as GOOD

trial 13 marked as GOOD

trial 14 marked as GOOD

trial 14 marked as GOOD

trial 15 marked as GOOD

trial 15 marked as GOOD

trial 16 marked as GOOD

trial 17 marked as GOOD

trial 18 marked as GOOD

trial 18 marked as GOOD

trial 19 marked as GOOD

trial 19 marked as GOOD

trial 20 marked as GOOD

trial 20 marked as GOOD

trial 21 marked as BAD

trial 21 marked as BAD

trial 21 marked as BAD

trial 21 marked as BAD

trial 21 marked as BAD

trial 21 marked as BAD

trial 21 marked as BAD

trial 21 marked as BAD

trial 21 marked as BAD

trial 21 marked as BAD

trial 21 marked as BAD

trial 21 marked as BAD

trial 21 marked as GOOD

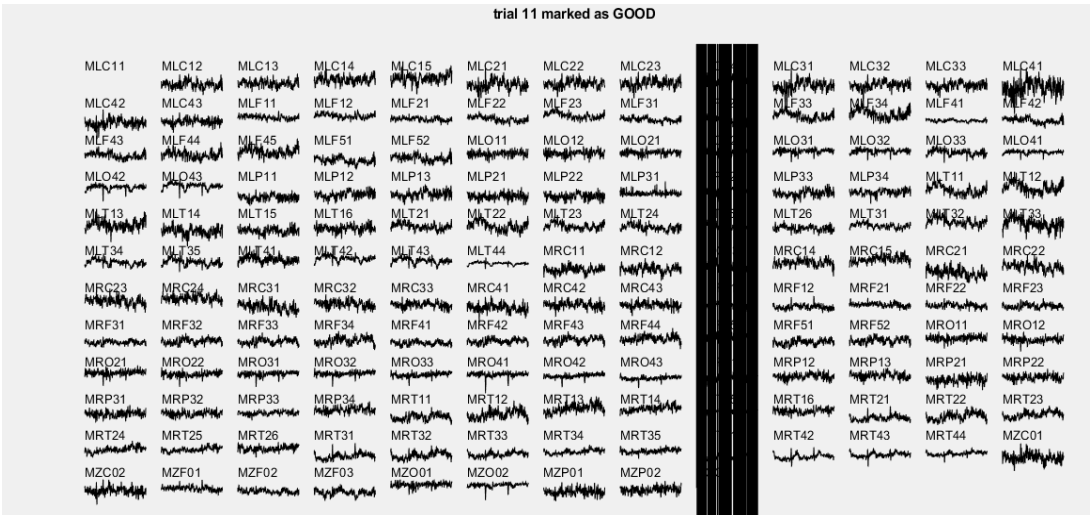
trial 21 marked as GOOD

trial 21 marked as GOOD

trial 21 marked as BAD

trial 21 marked as BAD

trial 21 marked as GOOD

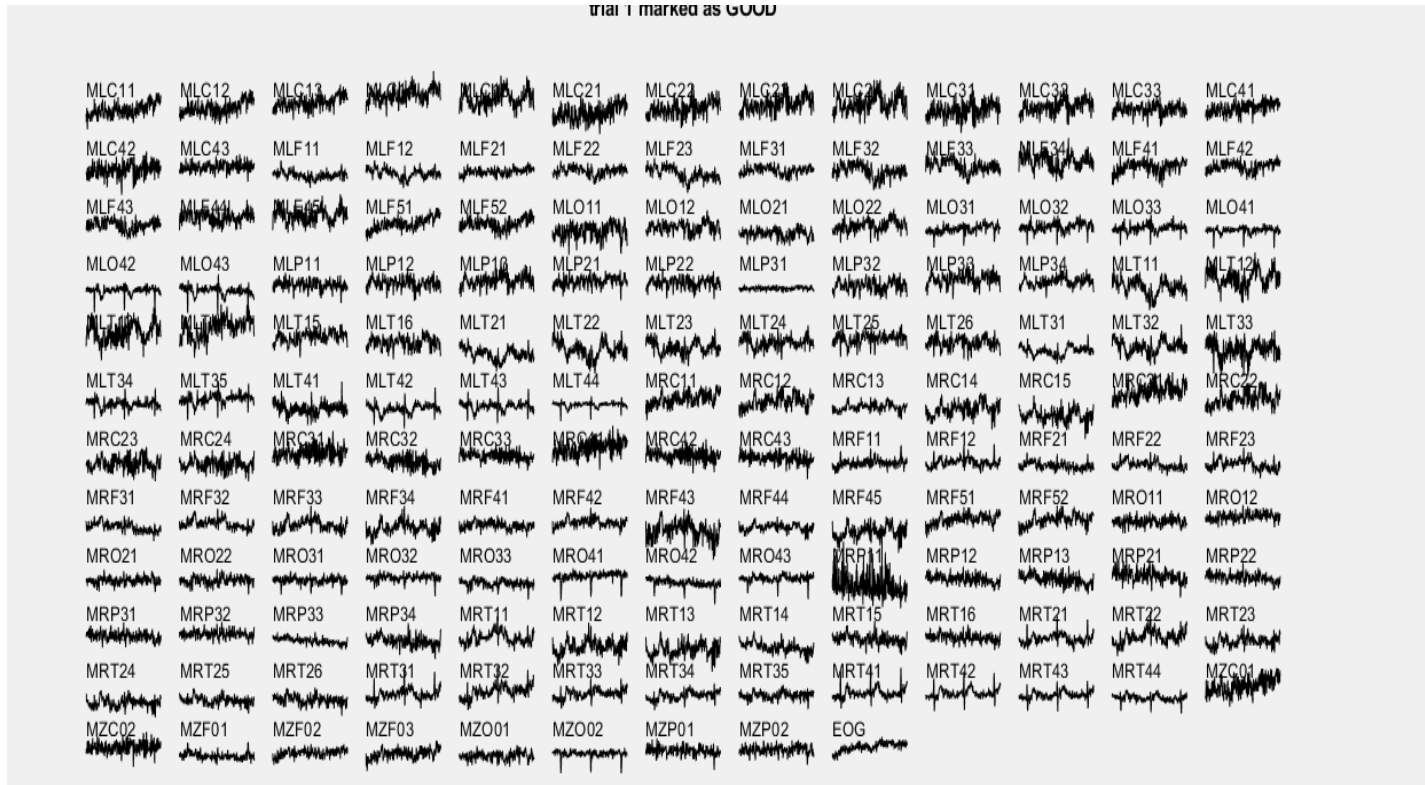


the input is raw data with 152 channels and 87 trials

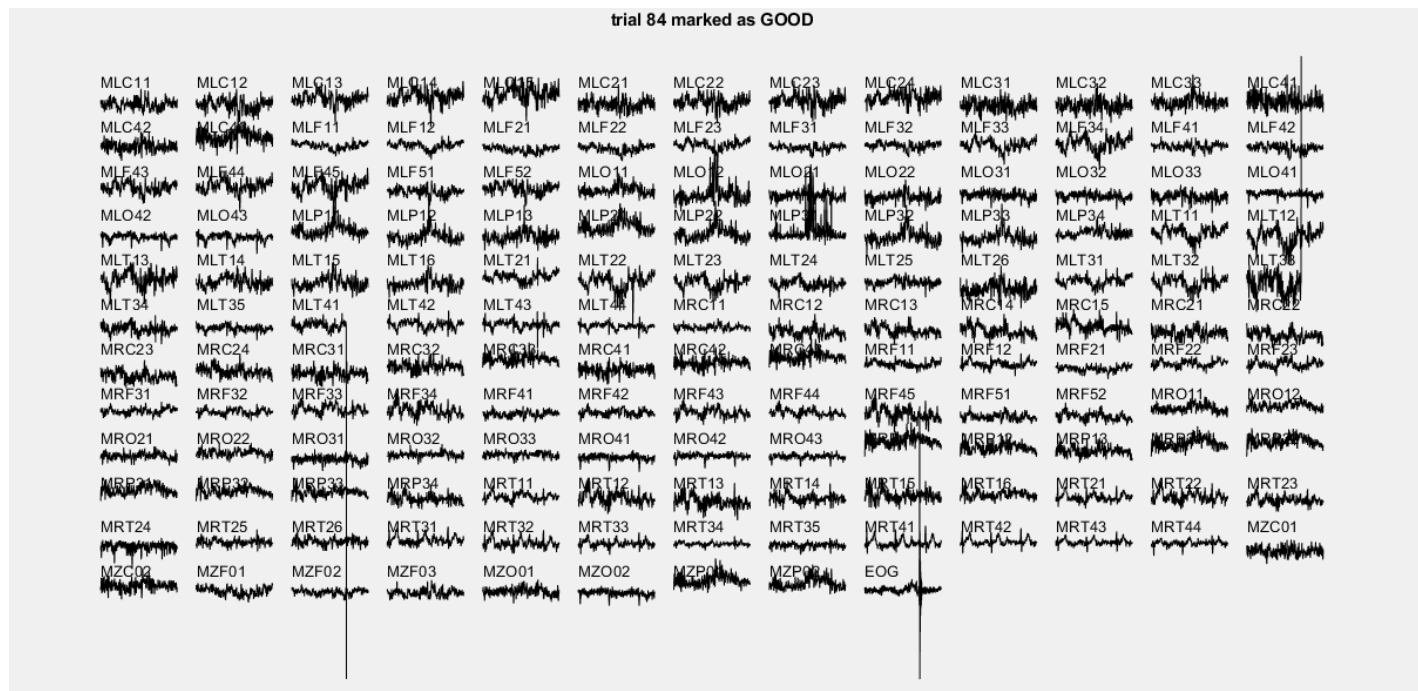
the call to "ft_selectdata" took 0 seconds

showing the scaled per trial, all channels at once

trial 1 marked as GOOD



Trial 84 shows an artifact which is caused by the electronics. Notice the jump in sensor MLC41.



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.

There are 87 trials in your data and first you reject trial 15, 36 and 39. Then trial number 87 becomes trial number 84. Later when you also want to reject trials 42, 43, 45, 49, 50, 81, 82 and 84 you should be very careful and subtract 3 from all the old trial numbers. If you would like to know which trials you rejected, it is best to call `ft_rejectvisual` only once.

Manual artifact rejection - display one channel at a time

It can also be convenient to view data from one channel at a time. This can be particularly relevant for the EOG channel.

```
cfg = [];
cfg.method = 'channel';
cfg.alim = 1e-12;
cfg.megscale = 1;
cfg.eogscale = 5e-8;
dummy = ft_rejectvisual(cfg, dataFIC);
showing the scaled data per channel, all trials at once
```

channel MLC11 marked as GOOD

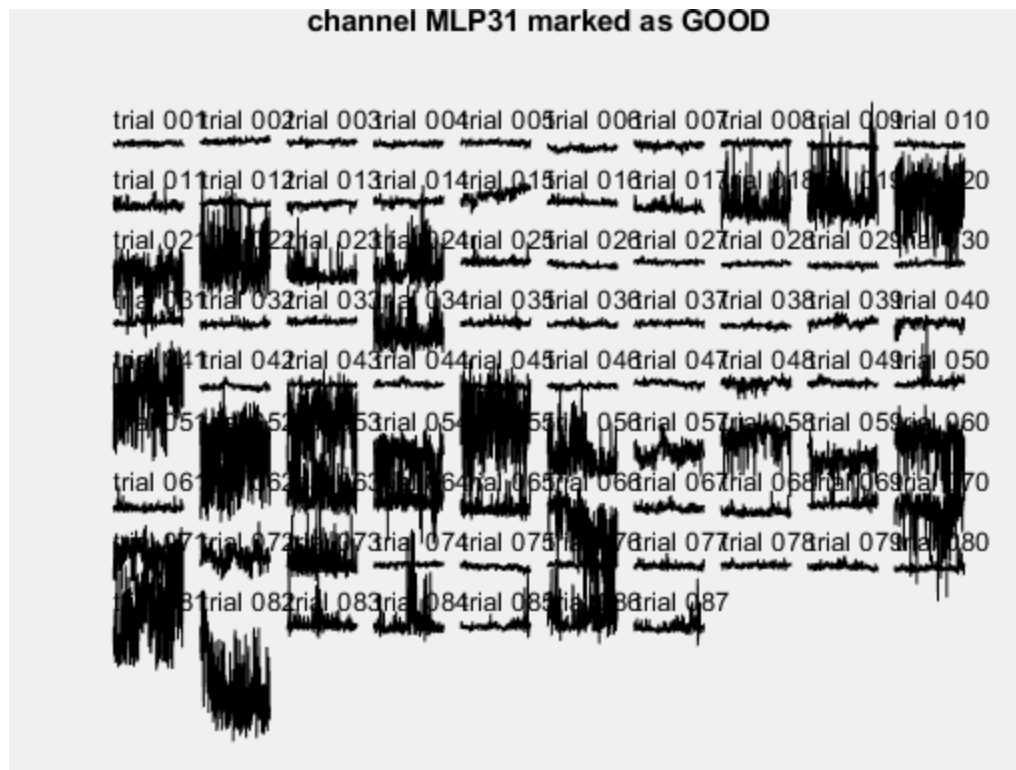
channel MLC11 marked as GOOD

trial 001trial 002trial 003trial 004trial 005trial 006trial 007trial 008trial 009trial 010
trial 011trial 012trial 013trial 014trial 015trial 016trial 017trial 018trial 019trial 020
trial 021trial 022trial 023trial 024trial 025trial 026trial 027trial 028trial 029trial 030
trial 031trial 032trial 033trial 034trial 035trial 036trial 037trial 038trial 039trial 040
trial 041trial 042trial 043trial 044trial 045trial 046trial 047trial 048trial 049trial 050
trial 051trial 052trial 053trial 054trial 055trial 056trial 057trial 058trial 059trial 060
trial 061trial 062trial 063trial 064trial 065trial 066trial 067trial 068trial 069trial 070
trial 071trial 072trial 073trial 074trial 075trial 076trial 077trial 078trial 079trial 080
trial 081trial 082trial 083trial 084trial 085trial 086trial 087

Click through the data using the > button. While clicking through all the trials you see that channels MLO12 and MLP31 contain a lot of artifacts (see the figure below). They should be marked as 'bad'. After pressing the 'quit' button the channels marked 'bad' are now removed from the data structure.

channel MLO12 marked as BAD

trial 001trial 002trial 003trial 004trial 005trial 006trial 007trial 008trial 009trial 010
trial 011trial 012trial 013trial 014trial 015trial 016trial 017trial 018trial 019trial 020
trial 021trial 022trial 023trial 024trial 025trial 026trial 027trial 028trial 029trial 030
trial 031trial 032trial 033trial 034trial 035trial 036trial 037trial 038trial 039trial 040
trial 041trial 042trial 043trial 044trial 045trial 046trial 047trial 048trial 049trial 050
trial 051trial 052trial 053trial 054trial 055trial 056trial 057trial 058trial 059trial 060
trial 061trial 062trial 063trial 064trial 065trial 066trial 067trial 068trial 069trial 070
trial 071trial 072trial 073trial 074trial 075trial 076trial 077trial 078trial 079trial 080
trial 081trial 082trial 083trial 084trial 085trial 086trial 087



Manual artifact rejection - display a summary

To produce an overview of the data choose the `cfg.method` 'summary

```
cfg = [];
cfg.method = 'summary';
cfg.alim = 1e-12;
dummy = ft_rejectvisual(cfg,dataFIC);
```

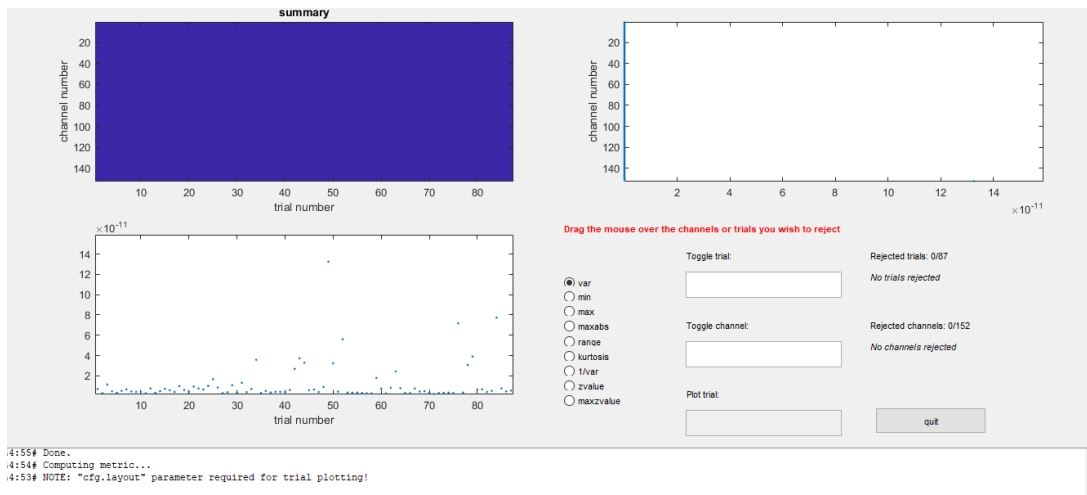
the input is raw data with 152 channels and 87 trials

the call to "ft_selectdata" took 0 seconds

showing a summary of the data for all channels and trials

```
computing metric [-]                                ]computing metric [-----/
]computing metric [-----
```

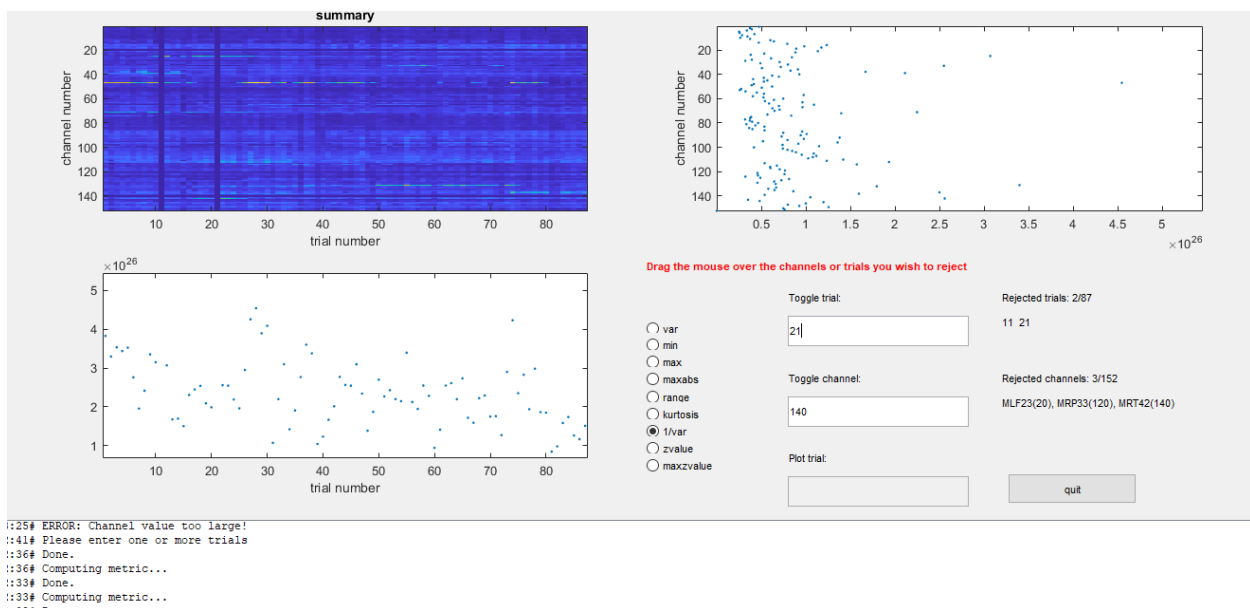
This gives you a plot with the variance for each channel and trial.



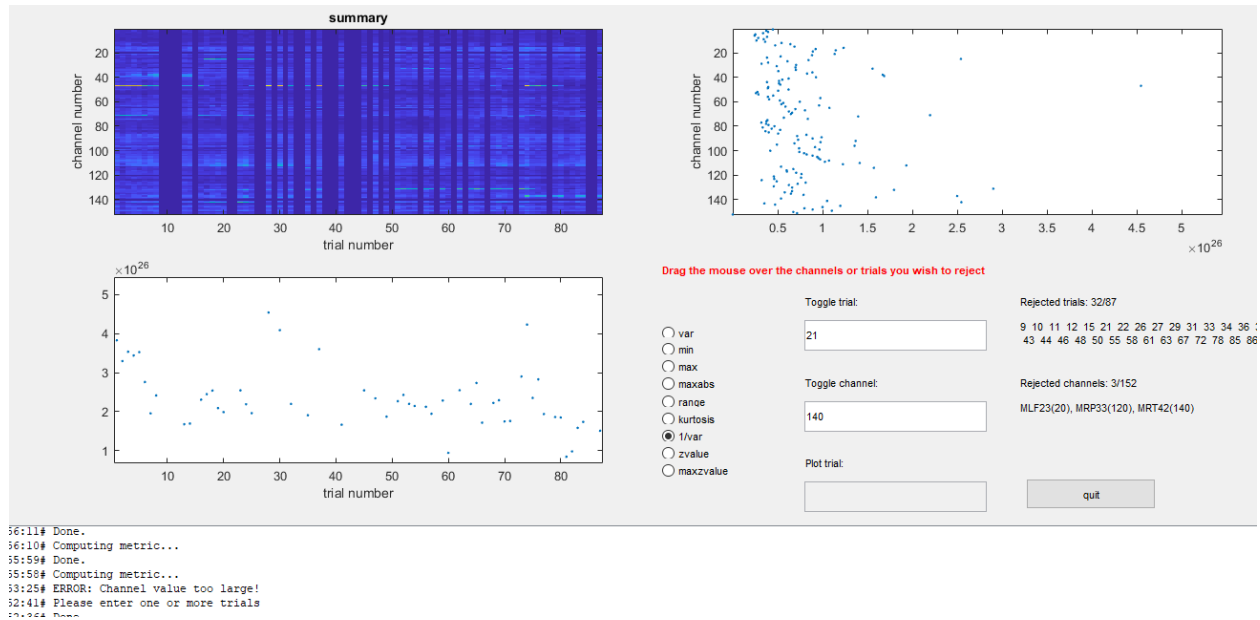
You should note that there is one channel which has a very high variance. That is the EOG channel, which contains numbers in uV which are of a very different order of magnitude than all MEG channels in T. Toggling the EOG channel will also change the figure with the maximal variance per trial (second row, left) a lot. Then you only see the variance in each trial in the MEG channels.

The command window allows for toggling trials and channels on and off.

Before pressing the 'Quit' button, you can always toggle the channels/trials back on, by using the edit boxes 'Toggle trial' or 'Toggle channel'.



After drading the mouse on channel.rejecting the trail and channel according to the requirement



After quitting, the trials/channels will be rejected from the data set and the command line output appears as follow

55 trials marked as GOOD, 32 trials marked as BAD

149 channels marked as GOOD, 3 channels marked as BAD

the following channels were removed: MLF23, MRP33, MRT42

the following trials were removed: 9, 10, 11, 12, 15, 21, 22, 26, 27, 29, 31, 33, 34, 36, 38, 39, 40, 42, 43, 44, 46, 48, 50, 55, 58, 61, 63, 67, 72, 78, 85, 86

the call to "ft_selectdata" took 0 seconds

the call to "ft_rejectvisual" took 910 seconds

This operation could be repeated for each of the metrics, by clicking on the different radio buttons 'var', 'min', 'max', etc.

he summary mode in **ft_rejectvisual** has been primarily designed to visually screen for artefacts in channels of a consistent type, i.e. in this example only for the axial MEG gradiometers.

You can use the following options in **ft_rejectvisual** to apply a scaling to the channels prior to visualisation: *cfg.eegscale*, *cfg.eogscale*, *cfg.ecgscale*, *cfg.emgscale*, *cfg.megscale*, *cfg.gradscale* and *cfg.magscale*

You can also call **ft_rejectvisual** multiple times, once for every type of channels in your data. If you use `cfg.keepchannel='yes'`, channels will not be removed from the data on the subsequent calls. For example:

```
cfg = [];
cfg.method = 'summary';
cfg.keepchannel = 'yes';

cfg.channel = 'MEGMAG';
clean1 = ft_rejectvisual(cfg, orig);

cfg.channel = 'MEGGRAD';
clean2 = ft_rejectvisual(cfg, clean1);

cfg.channel = 'EEG';
clean3 = ft_rejectvisual(cfg, clean2);
```

You can repeat this for the initially congruent (IC) condition. To detect all the artifacts use **ft_rejectvisual** with all 3 methods (trial, channel and summary) like in the examples above.

```
clear all
load PreprocData dataIC

cfg = [];
cfg.method = 'trial'; % also try cfg.method = 'channel' and cfg.method = 'summary'
cfg.alim = 1e-12;
cfg.megscale = 1;
cfg.eogscale = 5e-8;
dummy = ft_rejectvisual(cfg,dataIC);
```

trial 1 marked as GOOD

```
MLC1MLC12MLC13MLC14MLC15MLC16MLC17MLC18MLC19MLC20MLC21MLC22MLC23MLC24MLC25MLC26MLC27MLC28MLC29MLC30MLC31MLC32MLC33MLC34MLC35MLC36MLC37MLC38MLC39MLC40MLC41
MLC42MLC43MLC44MLC45MLC46MLC47MLC48MLC49MLC50MLC51MLC52MLC53MLC54MLC55MLC56MLC57MLC58MLC59MLC60MLC61MLC62MLC63MLC64MLC65MLC66MLC67MLC68MLC69MLC70MLC71MLC72MLC73MLC74MLC75MLC76MLC77MLC78MLC79MLC80MLC81MLC82MLC83MLC84MLC85MLC86MLC87MLC88MLC89MLC90MLC91MLC92MLC93MLC94MLC95MLC96MLC97MLC98MLC99MLC100
MLF1MLF2MLF3MLF4MLF5MLF6MLF7MLF8MLF9MLF10MLF11MLF12MLF13MLF14MLF15MLF16MLF17MLF18MLF19MLF20MLF21MLF22MLF23MLF24MLF25MLF26MLF27MLF28MLF29MLF30MLF31MLF32MLF33MLF34MLF35MLF36MLF37MLF38MLF39MLF40MLF41MLF42MLF43MLF44MLF45MLF46MLF47MLF48MLF49MLF50MLF51MLF52MLF53MLF54MLF55MLF56MLF57MLF58MLF59MLF60MLF61MLF62MLF63MLF64MLF65MLF66MLF67MLF68MLF69MLF70MLF71MLF72MLF73MLF74MLF75MLF76MLF77MLF78MLF79MLF80MLF81MLF82MLF83MLF84MLF85MLF86MLF87MLF88MLF89MLF90MLF91MLF92MLF93MLF94MLF95MLF96MLF97MLF98MLF99MLF100
MLO1MLO2MLO3MLO4MLO5MLO6MLO7MLO8MLO9MLO10MLO11MLO12MLO13MLO14MLO15MLO16MLO17MLO18MLO19MLO20MLO21MLO22MLO23MLO24MLO25MLO26MLO27MLO28MLO29MLO30MLO31MLO32MLO33MLO34MLO35MLO36MLO37MLO38MLO39MLO40MLO41MLO42MLO43MLO44MLO45MLO46MLO47MLO48MLO49MLO50MLO51MLO52MLO53MLO54MLO55MLO56MLO57MLO58MLO59MLO60MLO61MLO62MLO63MLO64MLO65MLO66MLO67MLO68MLO69MLO70MLO71MLO72MLO73MLO74MLO75MLO76MLO77MLO78MLO79MLO80MLO81MLO82MLO83MLO84MLO85MLO86MLO87MLO88MLO89MLO90MLO91MLO92MLO93MLO94MLO95MLO96MLO97MLO98MLO99MLO100
MLT1MLT2MLT3MLT4MLT5MLT6MLT7MLT8MLT9MLT10MLT11MLT12MLT13MLT14MLT15MLT16MLT17MLT18MLT19MLT20MLT21MLT22MLT23MLT24MLT25MLT26MLT27MLT28MLT29MLT30MLT31MLT32MLT33MLT34MLT35MLT36MLT37MLT38MLT39MLT40MLT41MLT42MLT43MLT44MLT45MLT46MLT47MLT48MLT49MLT50MLT51MLT52MLT53MLT54MLT55MLT56MLT57MLT58MLT59MLT60MLT61MLT62MLT63MLT64MLT65MLT66MLT67MLT68MLT69MLT70MLT71MLT72MLT73MLT74MLT75MLT76MLT77MLT78MLT79MLT80MLT81MLT82MLT83MLT84MLT85MLT86MLT87MLT88MLT89MLT90MLT91MLT92MLT93MLT94MLT95MLT96MLT97MLT98MLT99MLT100
MRC1MRC2MRC3MRC4MRC5MRC6MRC7MRC8MRC9MRC10MRC11MRC12MRC13MRC14MRC15MRC16MRC17MRC18MRC19MRC20MRC21MRC22MRC23MRC24MRC25MRC26MRC27MRC28MRC29MRC30MRC31MRC32MRC33MRC34MRC35MRC36MRC37MRC38MRC39MRC40MRC41MRC42MRC43MRC44MRC45MRC46MRC47MRC48MRC49MRC50MRC51MRC52MRC53MRC54MRC55MRC56MRC57MRC58MRC59MRC60MRC61MRC62MRC63MRC64MRC65MRC66MRC67MRC68MRC69MRC70MRC71MRC72MRC73MRC74MRC75MRC76MRC77MRC78MRC79MRC80MRC81MRC82MRC83MRC84MRC85MRC86MRC87MRC88MRC89MRC90MRC91MRC92MRC93MRC94MRC95MRC96MRC97MRC98MRC99MRC100
MRO1MRO2MRO3MRO4MRO5MRO6MRO7MRO8MRO9MRO10MRO11MRO12MRO13MRO14MRO15MRO16MRO17MRO18MRO19MRO20MRO21MRO22MRO23MRO24MRO25MRO26MRO27MRO28MRO29MRO30MRO31MRO32MRO33MRO34MRO35MRO36MRO37MRO38MRO39MRO40MRO41MRO42MRO43MRO44MRO45MRO46MRO47MRO48MRO49MRO50MRO51MRO52MRO53MRO54MRO55MRO56MRO57MRO58MRO59MRO60MRO61MRO62MRO63MRO64MRO65MRO66MRO67MRO68MRO69MRO70MRO71MRO72MRO73MRO74MRO75MRO76MRO77MRO78MRO79MRO80MRO81MRO82MRO83MRO84MRO85MRO86MRO87MRO88MRO89MRO90MRO91MRO92MRO93MRO94MRO95MRO96MRO97MRO98MRO99MRO100
MRP1MRP2MRP3MRP4MRP5MRP6MRP7MRP8MRP9MRP10MRP11MRP12MRP13MRP14MRP15MRP16MRP17MRP18MRP19MRP20MRP21MRP22MRP23MRP24MRP25MRP26MRP27MRP28MRP29MRP30MRP31MRP32MRP33MRP34MRP35MRP36MRP37MRP38MRP39MRP40MRP41MRP42MRP43MRP44MRP45MRP46MRP47MRP48MRP49MRP50MRP51MRP52MRP53MRP54MRP55MRP56MRP57MRP58MRP59MRP60MRP61MRP62MRP63MRP64MRP65MRP66MRP67MRP68MRP69MRP70MRP71MRP72MRP73MRP74MRP75MRP76MRP77MRP78MRP79MRP80MRP81MRP82MRP83MRP84MRP85MRP86MRP87MRP88MRP89MRP90MRP91MRP92MRP93MRP94MRP95MRP96MRP97MRP98MRP99MRP100
MRT1MRT2MRT3MRT4MRT5MRT6MRT7MRT8MRT9MRT10MRT11MRT12MRT13MRT14MRT15MRT16MRT17MRT18MRT19MRT20MRT21MRT22MRT23MRT24MRT25MRT26MRT27MRT28MRT29MRT30MRT31MRT32MRT33MRT34MRT35MRT36MRT37MRT38MRT39MRT40MRT41MRT42MRT43MRT44MRT45MRT46MRT47MRT48MRT49MRT50MRT51MRT52MRT53MRT54MRT55MRT56MRT57MRT58MRT59MRT60MRT61MRT62MRT63MRT64MRT65MRT66MRT67MRT68MRT69MRT70MRT71MRT72MRT73MRT74MRT75MRT76MRT77MRT78MRT79MRT80MRT81MRT82MRT83MRT84MRT85MRT86MRT87MRT88MRT89MRT90MRT91MRT92MRT93MRT94MRT95MRT96MRT97MRT98MRT99MRT100
MZC01MZC02MZC03MZC04MZC05MZC06MZC07MZC08MZC09MZC10MZC11MZC12MZC13MZC14MZC15MZC16MZC17MZC18MZC19MZC20MZC21MZC22MZC23MZC24MZC25MZC26MZC27MZC28MZC29MZC30MZC31MZC32MZC33MZC34MZC35MZC36MZC37MZC38MZC39MZC40MZC41MZC42MZC43MZC44MZC45MZC46MZC47MZC48MZC49MZC50MZC51MZC52MZC53MZC54MZC55MZC56MZC57MZC58MZC59MZC60MZC61MZC62MZC63MZC64MZC65MZC66MZC67MZC68MZC69MZC70MZC71MZC72MZC73MZC74MZC75MZC76MZC77MZC78MZC79MZC80MZC81MZC82MZC83MZC84MZC85MZC86MZC87MZC88MZC89MZC90MZC91MZC92MZC93MZC94MZC95MZC96MZC97MZC98MZC99MZC100
MZOG
```

: < > << >> bad good bad> good>

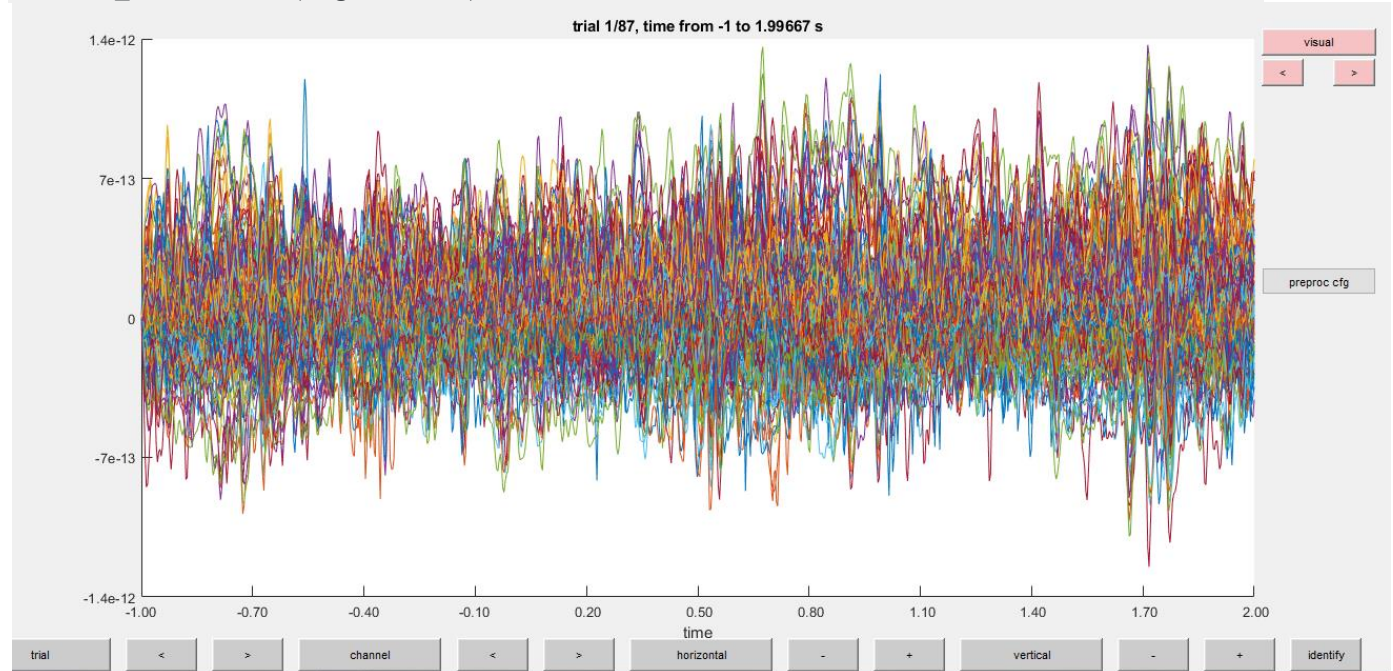
Trials 1, 2, 3, 4, 14, 15, 16, 17, 20, 35, 39, 40, 47, 78, 79, 80, 86 contain various artifacts, classify these as 'BAD'. Also reject the channels MLO12 and MLP31.

That already done previously.

Use `ft_databrowser` to mark the artifacts manually

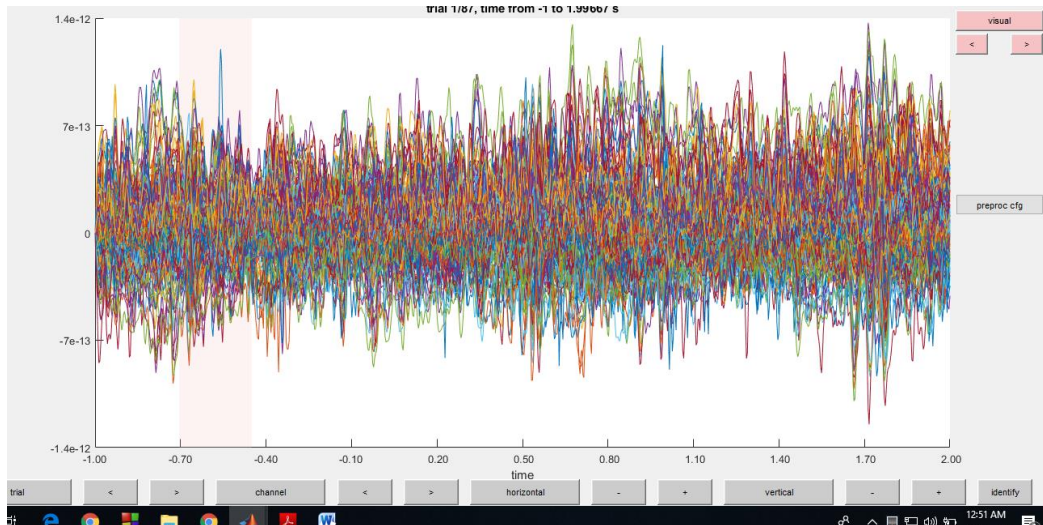
An alternative way to remove artifacts is to page through the butterfly plots of the single trials, by using the `ft_databrowser` function. Call the function like

```
% first select only the MEG channels
cfg = [];
cfg.channel = 'MEG';
data = ft_preprocessing(cfg,dataFIC);
% open the browser and page through the trials
cfg=[];
cfg.channel = 'MEG';
artf=ft_databrowser(cfg,dataFIC);
```

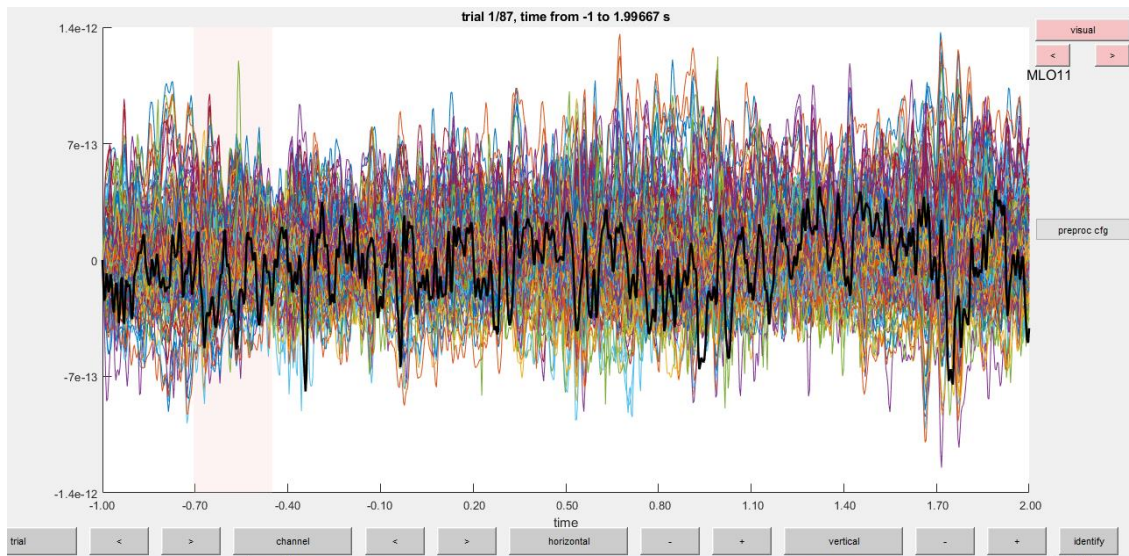


The resulting variable contains the file

```
artf.artfctdef.visual.artifact = [begartf endartf]
```

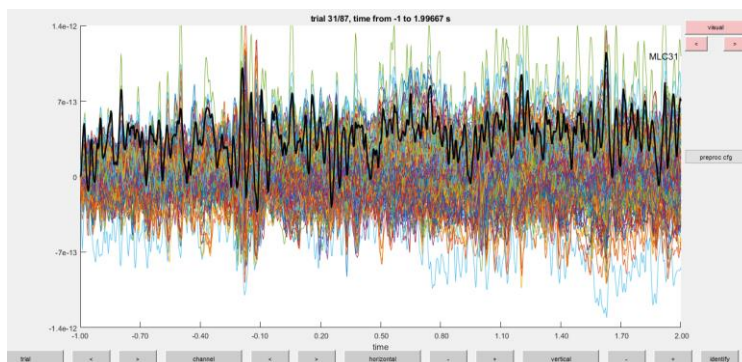


The artifact is detected and dragged by mouse clearly shown in graph.



I understood that this graph is summation of all the channel data in first trail.

We can change the trail and can detect the artifacts manually and select it and remove it.



Summary

Per condition, the following trials contain artifacts:

- FIC: 15, 36, 39, 42, 43, 49, 50, 81, 82, 84
- IC: 1, 2, 3, 4, 14, 15, 16, 17, 20, 35, 39, 40, 47, 78, 79, 80, 86
- FC: 2, 3, 4, 30, 39, 40, 41, 45, 46, 47, 51, 53, 59, 77, 85
- Channels MLO12 and MLP31 are removed because of artifacts.

Automatic artifact rejection

Automatic artifact rejection in FieldTrip is a sophisticated and complicated approach to artifact rejection

Introduction

Before further analysis in any of the other tutorials, it is best to have artifact free data. Within FieldTrip you can choose to do visual/manual or automatic artifact detection and rejection.

Background

For a successful analysis of EEG or MEG signals, “clean” data is required. That means that you should try to reduce variance in the data due to factors unrelated to your experimental conditions.

Procedure

The following steps are used to detect artifacts in FieldTrip’s automatic artifact rejection (see figure below)

1. Defining segments of interest using `ft_definetrial`
2. Detecting artifacts using `ft_artifact_zvalue`, this consists of
 - Reading the data (with padding) from disk
 - Filtering the data
 - Z-transforming the filtered data and averaging it over channels
 - Threshold the accumulated z-score

Artifact definition a two column array with the onset and offset sample number of every detected artifact.

I. Reading the data (with padding) from disk

Physiological data is often very large and takes system memory. For this reason it is most efficient to read only the data you need from disk instead of reading in all the data first and selecting interesting segments later.

Read the data and determine the artifacts.

Try to remove the artifacts from the data.

as in all the cases where we use trial- and filter-padding more data is needed, these cannot be applied and you will become much more sensitive to e.g. filter issues. This will also be an issue when data is not recorded continuously.

II. Filtering the data

We can optimize filter settings for EOG and SQUID jump artifacts.

III. Z-transforming the filtered data and averaging it over channels

Artifact detection approach-

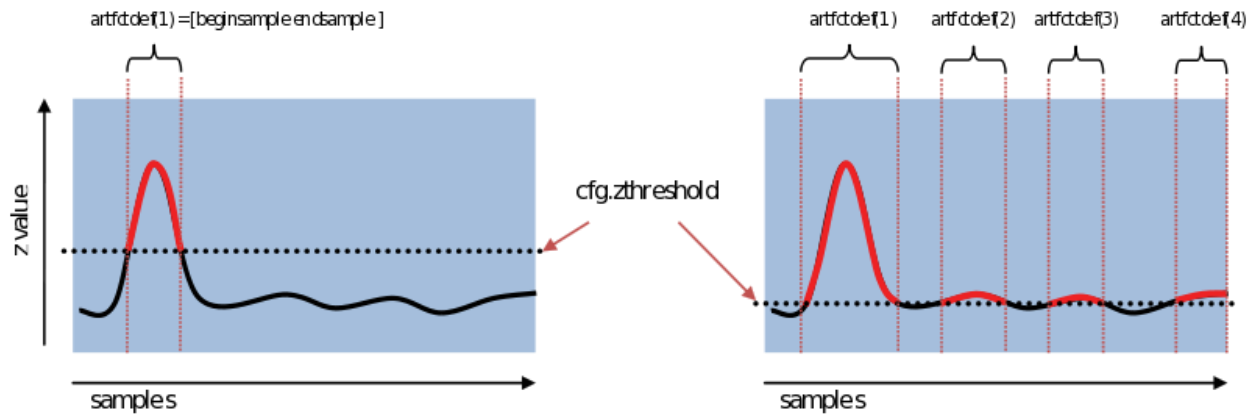
- 1) they occur sparsely in time, i.e. not all the time.
- 2) they are reflected by larger amplitudes than the brain data (possibly or especially in a particular frequency band).
- 3) they occur over multiple channels/electrodes.

After Filtering ,transformed

1. Per channel/electrode the amplitude of the signal over time is calculated (the Hilbert envelope)
2. Per channel/electrode its mean and standard deviation is calculated (over all samples)
3. Per channel/electrode every timepoint is z-normalized (mean subtracted and divided by standard deviation)

4. Per timepoint these z-values are averaged. Since an artifact might occur on any and often on more than one electrode (think of eyeblinks and muscle artifacts), averaging z-values over channels/electrodes allows evidence for an artifact to accumulate. *This results in one timecourse representing standardized deviations from the mean of all channels.*

IV. Thresholding the accumulated z-score



Setting a higher or lower z-value threshold will make the detection of artifacts more conservative or liberal.

Padding

Artifact padding

("cfg.artfctdef.xxx.artpadding") is used

Trial padding

("cfg.artfctdef.xxx.trlpadding") is used

Filter padding

("cfg.artfctdef.xxx.fltpadding") is used

Artifact rejection

First we need to define our trial

```

cfg                = [];
cfg.dataset        = 'ArtifactMEG.ds';
cfg.headerformat   = 'ctf_ds';
cfg.dataformat     = 'ctf_ds';
cfg.trialdef.eventtype = 'trial';
cfg                = ft_definetrial(cfg);
trl                = cfg.trl(1:50,:); % we'll only use the first 50 trials for
this example

```

getRawCTFBalanceCoefs : Sensor Q11 appears in ds.res4.scr, but not in ds.res4.chanNames

Jump artifact detection

For detecting jump artifacts, begin with the following parameter

```

% jump
cfg = [];
cfg.trl = trl;
cfg.datafile = 'ArtifactMEG.ds';
cfg.headerfile = 'ArtifactMEG.ds';
cfg.continuous = 'yes';

% channel selection, cutoff and padding
cfg.artfctdef.zvalue.channel = 'MEG';
cfg.artfctdef.zvalue.cutoff = 20;
cfg.artfctdef.zvalue.trlpadding = 0;
cfg.artfctdef.zvalue.artpadding = 0;
cfg.artfctdef.zvalue.fltpadding = 0;

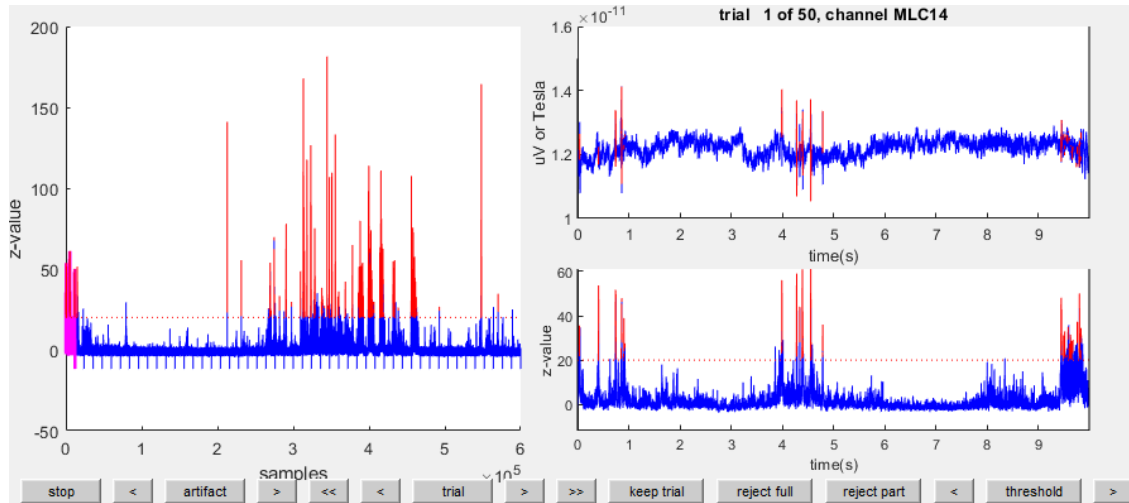
% algorithmic parameters
cfg.artfctdef.zvalue.cumulative = 'yes';
cfg.artfctdef.zvalue.medianfilter = 'yes';
cfg.artfctdef.zvalue.medianfiltord = 9;
cfg.artfctdef.zvalue.absdiff = 'yes';

% make the process interactive
cfg.artfctdef.zvalue.interactive = 'yes';

[cfg, artifact_jump] = ft_artifact_zvalue(cfg);

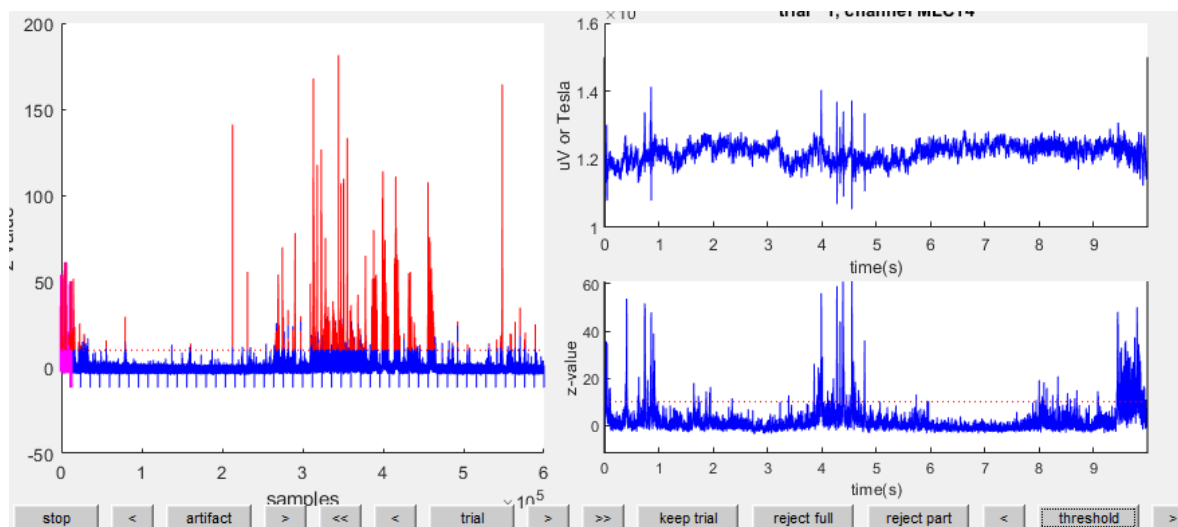
```

Interactive figure (below) of `ft_artifact_zvalue`. The left panel shows the z-score of the processed data, along with the threshold. Suprathreshold data points are marked in red. The lower right panel shows the z-score of the processed data for a particular trial, and the upper right panel shows the unprocessed data of the channel that contributed most to the (cumulated) z-score. You can browse through the trials using the buttons at the bottom of the figure. Also, you can adjust the threshold, and manually keep/reject trials.



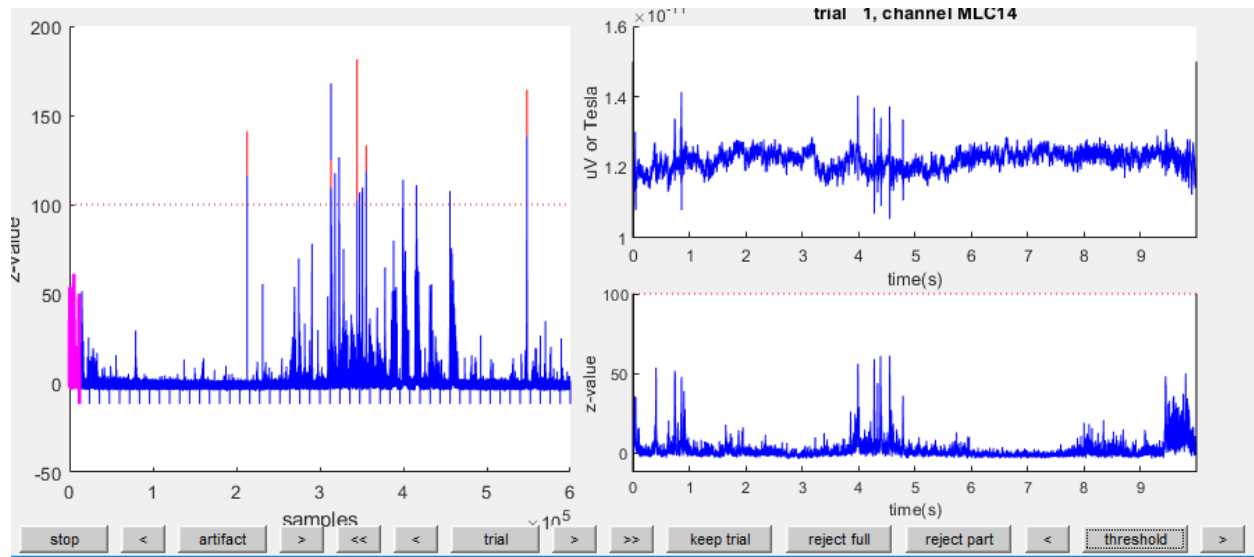
See how the graph is changing due to changing the z valve ,it reject the upper part.

Z value =10

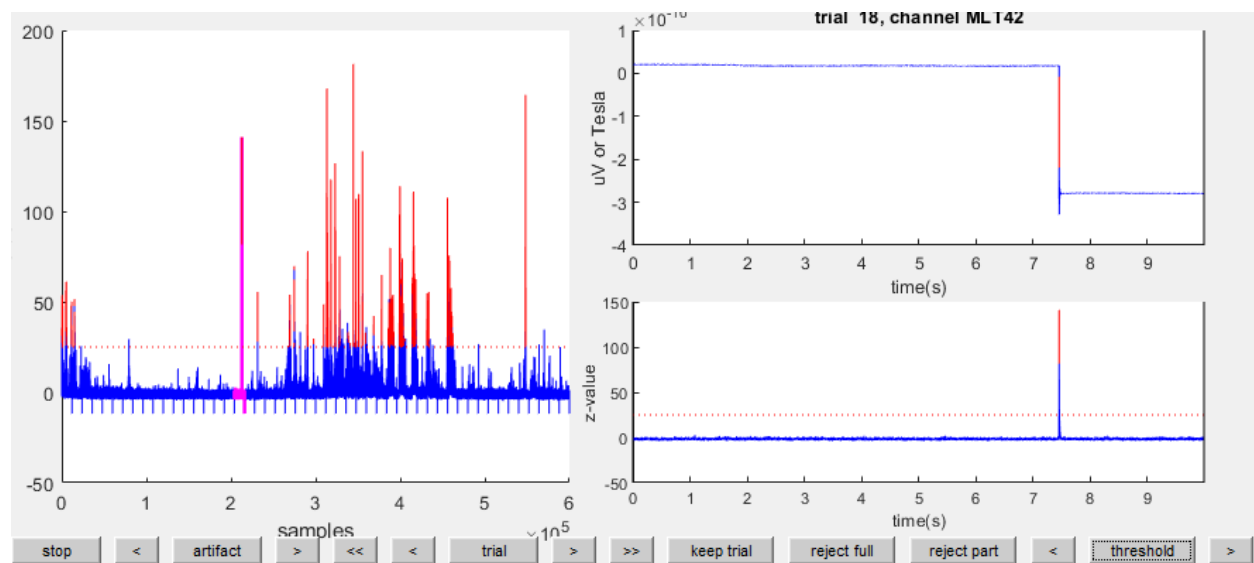


a lot of data points are detected to be a 'jump' artifact, although they seem more often 'muscular' in nature.

Z_value is 100 .



A typical SQUID bump can be seen in trail 18.



Detection of muscle artifacts

The same way as **ft_artifact_zvalue** is used to detect jump artifacts, it can be used to detect muscle artifacts.

```
% muscle
cfg = [];
cfg.trl = trl;
cfg.datafile = 'ArtifactMEG.ds';
cfg.headerfile = 'ArtifactMEG.ds';
cfg.continuous = 'yes';
```

```

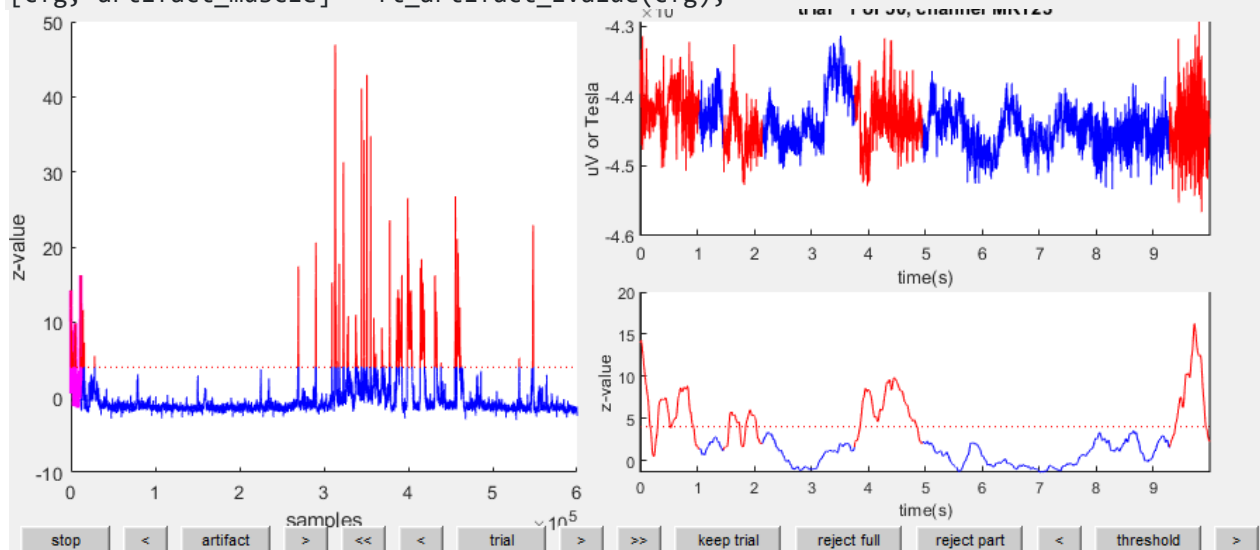
% channel selection, cutoff and padding
cfg.artfctdef.zvalue.channel    = 'MRT*';
cfg.artfctdef.zvalue.cutoff     = 4;
cfg.artfctdef.zvalue.trlpadding = 0;
cfg.artfctdef.zvalue.fltpadding = 0;
cfg.artfctdef.zvalue.artpadding = 0.1;

% algorithmic parameters
cfg.artfctdef.zvalue.bpfilter    = 'yes';
cfg.artfctdef.zvalue.bpfreq     = [110 140];
cfg.artfctdef.zvalue.bpfilterord = 9;
cfg.artfctdef.zvalue.bpfilttype = 'but';
cfg.artfctdef.zvalue.hilbert     = 'yes';
cfg.artfctdef.zvalue.boxcar      = 0.2;

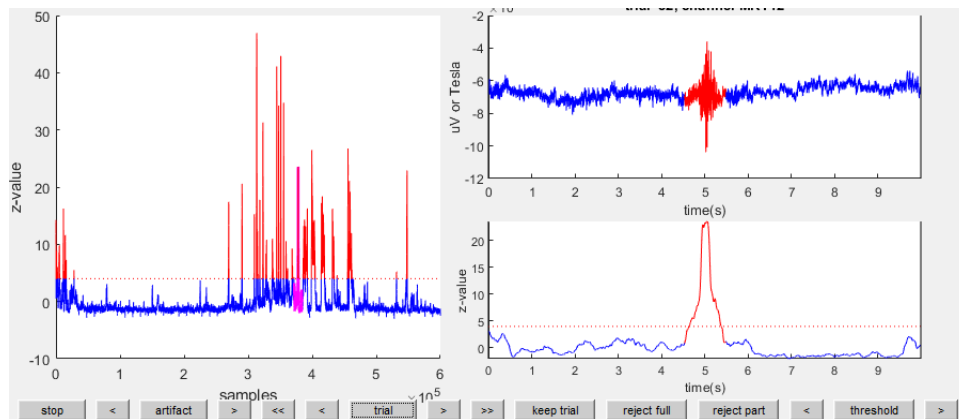
% make the process interactive
cfg.artfctdef.zvalue.interactive = 'yes';

[cfg, artifact_muscle] = ft_artifact_zvalue(cfg);

```



A typical muscle artifact can be observed on channel MRT12, trial 32.



Detection of EOG artifacts

The same way as **ft_artifact_zvalue** is used to detect jump artifacts, it can be used to detect eye blinks artifacts (EOG). Note that only the EOG is scanned in the eye artifacts case, which will take less time than scanning all MEG channels, which was needed for jump and muscle artifacts.

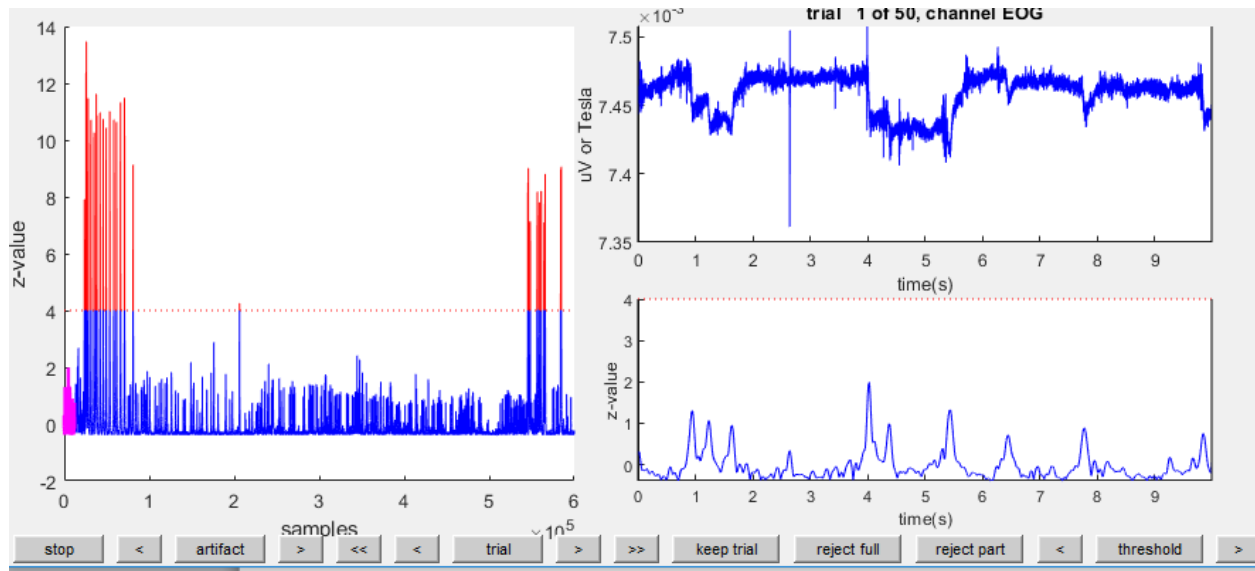
```
% EOG
cfg = [];
cfg.trl = trl;
cfg.datafile = 'ArtifactMEG.ds';
cfg.headerfile = 'ArtifactMEG.ds';
cfg.continuous = 'yes';

% channel selection, cutoff and padding
cfg.artfctdef.zvalue.channel = 'EOG';
cfg.artfctdef.zvalue.cutoff = 4;
cfg.artfctdef.zvalue.trlpadding = 0;
cfg.artfctdef.zvalue.artpadding = 0.1;
cfg.artfctdef.zvalue.fltpadding = 0;

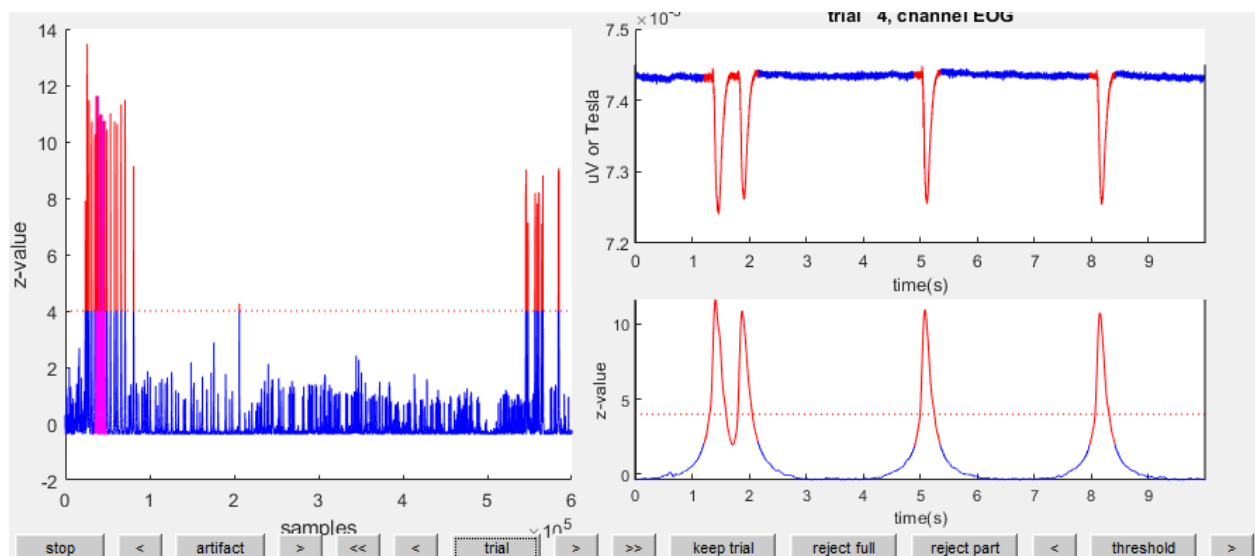
% algorithmic parameters
cfg.artfctdef.zvalue.bpfilter = 'yes';
cfg.artfctdef.zvalue.bpfilttype = 'but';
cfg.artfctdef.zvalue.bpfreq = [2 15];
cfg.artfctdef.zvalue.bpfiltord = 4;
cfg.artfctdef.zvalue.hilbert = 'yes';

% feedback
cfg.artfctdef.zvalue.interactive = 'yes';

[cfg, artifact_EOG] = ft_artifact_zvalue(cfg);
```



Typical eyeblink artifacts can be observed in trial 4. Note that the trial considers artifacts to precede and last longer than the threshold crossing, by padding data with 0.1 seconds on the left and 0.1 sec on the right



We are applying band pass filter to reject the unwanted signal.