# EAT data processing

# - README –

2. June 2022

Several scripts exist for data processing:

1. EAT\_settings.m
2. EAT\_processing.m
3. EAT\_merge.m
4. EAT\_smoothing\_force\_data.m
5. EAT\_segmentation\_force.m
6. EAT\_segmentation\_frequency.m
7. savedata.m

Prerequisites:

For the scripts to run without further modifications all data (task scripts/raw data/analysis scripts) need to be stored in a specific directory system that can be extended for different studies.

The main folder, for example, effort contains 3 subfolders

* + - 1. Analyses
      2. Data
      3. Project\_versions

In each of those folders a separate subfolder for each study should be there.

For example:

1. Analyses/TUE001
2. Data/TUE001
3. Project\_versions/TUE001

Additionally, in the Analyses folder a folder called general\_processing (or something comparable) should contain all scripts for the processing.

If your data is organized differently, you have to make the corresponding changes in EAT\_settings.m lines (100 – 127).

To run the analyses, **first run the EAT\_settings.m** script to create the settings. These are then existent in the workspace. For the settings, you need to set variables in the code or interact with the console. For details, see below. The settings are study specific and a corresponding .mat file will be saved in Analyses/\*Study\_name\*

When the settings are in the workspace, you can **then run EAT\_processing.m**. This will automatically process and segment your data, i.e. run the correct scripts for you. If you want to use settings that were previously saved, first load them to the workspace manually.

The following **dependencies** exist:

* *output data* must exist in Effort/Data/<study>  
  (for pilot data: Effort/Data/<study>/pilot)
* with the following generic name:  
  ExpEAT\_<study>\_<subj-id>\_S<sess-nr>\_R<run-nr> (experimental data)  
  TrainEAT\_<study>\_<subj-id>\_S<sess-nr>\_R<run-nr> (training data)

with <study>:

* + a **6**-character string containing the study name (e.g., TUE001)

with <subj-id>:

* + experimental:

in the end subject number

in the beginning padded with zeros so that length of 6

**! has to be 6 characters long!**

* + pilot:  
    in the beginning a 9

in the end the subject number

in-between padded with zeros so that it is of length 6

with <sess-nr>:

* + session ID between 1 and 9

with <run-nr>:

* + run ID between 1 and 9
* if *stimulation or group conditions* exist, this file must exist in Effort/Analyses/<study>  
  with generic name: <study>\_StimCond.mat

For example, a file containing group labels HC vs. MDD. See example file (TUE003\_cond.mat)

* file with *experimental settings* must exist in Effort/Project\_Versions/<study>  
  with generic name: EATsettings\_<study>\_ S<sess-nr>\_R<run-nr>

Study specific settings that are saved with the project code. There can be different settings for different sessions.

Files that are stored in a sub-folder won’t be found by MATLAB!

# **EAT\_settings.m**

# This script always asks you, if you want to set the variables by interacting with the console.

# Do you want to set the settings by interacting with the console? [0 = no, 1 = yes]

# If you answer 0, continue with the section “set variables manually”. For 1, continue with section “Interact with console”.

# **Set variables manually**

In the beginning of the script, there are several variables that you can set to your needs. Elsewise, you can interact with the console (see below).

|  |  |
| --- | --- |
| experiment.exp\_data | 1 = experimental data  0 = pilot data |
| process\_training | 1 = training data is processed as well  0 = training data is skipped |
| experiment.paradigm\_number | Save the string of the study name, example: ‘TUE005’ |

Participants for which data is not present, are excluded automatically.

If you want to exclude other participants, you can do that in l. 252.

**All available data from each participant is included also if there are (still) sessions missing**

**If you want to exclude participants with incomplete data they have to be excluded manually!**

# **Interact with console**

# Now, some specifications are needed about the data you want to process. If you are not sure whether you process experimental or pilot data, take a look at the coding scheme of the subject id (see above).

Experimental or pilot data? [0 = pilot, 1 = experimental]

Training data is usually not so interesting for analysis. In case you want to look at the training data nevertheless, you can specify that now.

Process training data? [0 = no, 1 = yes]

Now, the probably most important question follows. From which study do you process data? Your answer MUST have the following structure: TUE<study-nr> with <study-nr> has a length of 3, in the end is the study identifier and the beginning is padded with zeros. Enter without any extra signs (space, apostrophe, etc).

Please enter the paradigm number. [example: TUE004]

All participants of whom data is missing are excluded automatically. If only one session of many exist, the processing script can handle that. In case you have other reasons for exclusion, please enter that now.

Do you want to exclude participants from processing? [0 = no, 1 = yes]

After answering with 1, you can type the subject IDs that you want to exclude. Be careful! It must be written in a specific style. [ID1, ID2, ID3, …, IDN]

Which participant do you want to exclude? (Format: [id1,id2,...,idn])

In case the bracket format does not work for you, you can also enter single numbers (for example: 5). After every answer you see the question:

Do you want to exclude further participants from analysis? [0 = no, 1 = yes]

You can answer with 1 and enter all participants numbers you want to exclude one by one. If you are done, answer 0.

After that, you are done and you can see the confirmation:

Settings saved

Settings are saved in

TUE\_general/Tasks/Effort/Analyses/<study>

# **EAT\_processing.m**

# First, run the settings script or load settings. Then, run this script. After starting the script, every processing script is run automatically. Here is a flow chart, which scripts are run in which case:

Ein Bild, das Regenschirm enthält.

Automatisch generierte Beschreibung

# **EAT\_merge.m**

The RAW data comprises of one file per participant per session. The merge script is supposed to make one lucid and processable file out of the RAW data.

|  |  |
| --- | --- |
| l.42 | If you chose to process training data as well, runLabel\_start is set to 1, thus this loop runs from 1 (training data) to 2 (experimental data). Elsewise just experimental data. |
| l.44 | Run through all subjects. |
| ll. 47 – 53 | Printing to console if you choose that before. |
| l. 55 | Only continue if data is present (depending on your data exclusion choice) |
| ll. 61 – 71 | Create correct subject ID depending on coding scheme (9 is pilot data, 0 is experimental data) |
| ll. 73 – 101 | Read RAW data  → depending on run label, read training or exp data  → if training file is missing (for example session 2), script continues  → if experimental file is missing, the warning is printed to the console |
| ll. 105 – 121 | Output file - make table from matrix  Advantage: each column has a meaningful name to which can be referred in the code |
| ll. 123 – 127 | Read out number of trials, needed for segmentation (from each session) |
| ll. 129 – 143 | Derivations over time (1st and 2nd derivative and integral of RelEffort) |
| ll. 144 – 177 | Group and stimulation conditions are read out from the file (existent in Analyses/<study> folder) – this does not need to exist for every study! |
| ll. 178 – 313 | VAS ratings read out if VAS was done  → ll. 196 – 279: Some older studies have still other naming than the new ones, i.e. searching for different variable names  → ll. 281 – 313: ratings are read out trial-wise |
| ll. 326 - 408 | WOF |
| ll. 372 – 393 | Derivations of time |
| ll. 395 – 427 | End and saving of merging process Raw data was one file per subject Now we save on file for all subjects with named columns and added metrics (derivations) |

# **EAT\_smoothing\_force\_data.m**

The script adds one column RelForce\_RAW and overwrites RelForce. Using some formulas (look at Monja’s master thesis), the RelForce is smoothed. The non-smoothed, i.e. RAW, RelForce is saved in RelForce\_RAW. The smoothed version overwrites then the old data of RelForce. **If this script is used, all following scripts uses the smoothed data!** (Because this is saved in RelForce)

data is smoothed over one trial => start values of smoothing variables are re-set for every trial (ll. 78 - 88)

# **EAT\_segmentation\_force.m**

Segmentation workflow:

**For loops**

For each subject (l. 79) run through each session (l. 96) and each trial (l. 130). For each trial (l. 130) run through each time stamp, i.e. row (l. 140).

**Trial Segmentation**

To segment the data correctly, we see into the future with a sliding window. This window has the size of segmentation.windowSize.   
Currently, the value of 17 worked well. (l. 59)

An important value is the sum of all first derivatives over the sliding window COM10D1F (l. 160).

Two main processes are used to recognize the start of a new segment:

1. **Look at the big picture if new segment is possible**

l. 355 - main condition that change to work is likely

if COM10D1F > multiplicator \* STD

all other conditions optimize runtime or check special cases

l. 427 - main condition that change to rest is likely

if COM10D1F < - multiplicator \* STD

all other conditions optimize runtime or check special cases

l. 404 - for the beginning of the trial: special condition in case that the participant does not start with rest (default setting)

If nothing of this happens, continue last segment.

If the change to a new segment is likely, set two Booleans to 1:

(Work\_Ons and Work\_Prog) or (Rest\_Ons and Rest\_Prog)

The Ons(et) variables save the following: after a change to a new segment is likely, we want to know where exactly the new segment starts (T\_Work/T\_Rest) Until this exact starting point was found, this variable stays at value 1.

The Prog(ress) variables save the following: after a change to a new segment is likely and still after the time point was found, we wait with the change to the new segment until the threshold of 50% was reached. Thus, if this time point is reached, we set backwards the new segment (beginning with T\_Work/T\_Rest) until the current time point. If no new changes are recognized, this new segment continues automatically (see above: “If nothing of this happens, continue last segment.)

1. **Check details if a new segment really starts**

ll. 170 – 258 - T\_Work/T\_Rest is searched: if next time stamps in sliding window are all larger/smaller than the current time point, change the row as time point of segment start

ll. 259 – 345 - after time point was found, wait until 50% threshold and set the segment backwards

![Ein Bild, das Karte, Text enthält.

Automatisch generierte Beschreibung]()

![Ein Bild, das Karte enthält.

Automatisch generierte Beschreibung]()

![Ein Bild, das Karte, Text enthält.

Automatisch generierte Beschreibung]()