EEG Course: Semester project

P300/P3

A visual oddball experiment

by

Stefan Ott, Matrikel-Nr. 3563586

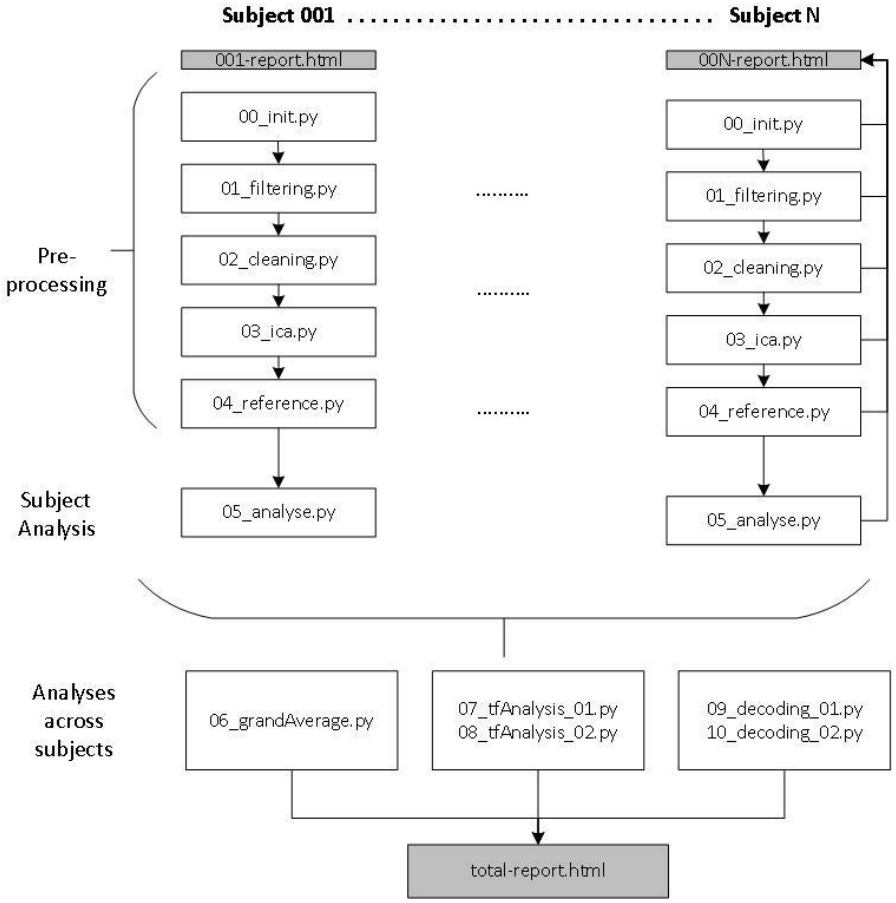
31.03.2021

Contents

Project description

1. Architecture:

1.1 Pipeline



The pipeline operates as describe in the picture above. In the pipeline each Python file is a script that either operates on single subjects or across multiple subjects. In the first stage Preprocessing/Subject Analysis is done for individual subjects. Later more advanced analysis techniques like the grandAverage peak analysis, time-frequency analysis and decoding-analysis are performed on the artefacts of multiple subjects. Steps produce plots for visual inspection of the data and analysis purposes. These are either saved in the individual subject reports, where each subject has its own HTML report file (using MNE.Report). Later analysis across subjects write in a combined total-report.html

The pipeline with several single python scripts can be execute together or individually. Besides the manual execution, the package **Pydoit**[1] is used to run the scripts automatically.
In general this pipeline follows the seven quick tips given by Marijn van Vliet [1]. Also some recommendations and best practices were reused over from his conpy[2] project like the **filenames class** which conveniently manages file paths. All credits for this work and the corresponding files stays of course with Marijn van Vliet[3]

[1] https://pydoit.org/
[2] https://github.com/AaltoImagingLanguage/conpy
[3] https://github.com/wmvanvliet

Using this has several advantages:

- **Data consistency/ Step-wise artefacts:**
  Each step takes as input the artefact of the previous step. Then it performs its operation and produces a new result. All the artefacts can be individually inspected. The pipeline also checks for existing already computed artefacts which then can be reused in the next run without recomputing all files from zero again.

- **Parallelization:**
  Pydoit takes fully care of parallelization of the specified jobs and uses the provided computing resource therefore very efficiently which saves time. Especially when computing results for multiple subjects this feature is quite useful.

- **Reproducibility:**
  The pipeline allows to conveniently reproduce the reported results in the correct order with the correct parameters. This is also supported by the individual **config.yaml** file which holds all important configuration settings and allows for easy adjustments.

- **File and Directory management**
  Automatic reports and file generation with the advanced file management system are very handy for analysis purposes. Everything is nicely structured in one place.

Pipeline execution:
The pipeline can be executed using Pydoit. The execution time is around ~40min. Needs ~10GB disk space (without ERP Core data).

parallel: `doit -n [nr. of threads] -P thread`
normal: `pydoit -m doit`
cleaning: `doit clean`

1.2. Config File

Quick information on the config files which holds all parameters used in the project. From these files the user can adjust most of the behavior of the system.

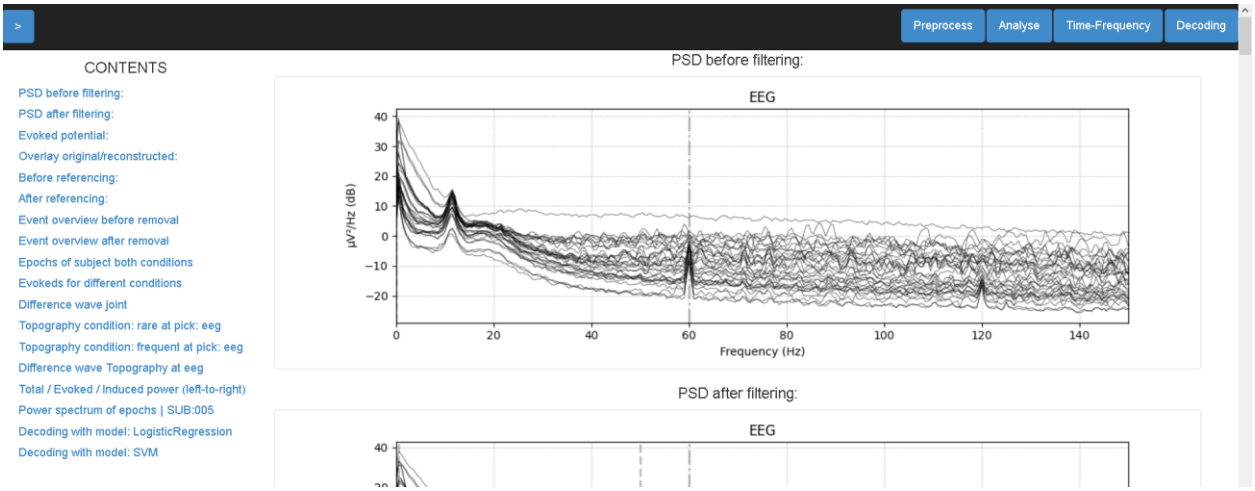
- Config.py:
  - Python file which manages filenames over fnames.py class from conpy[4]
  - Builds YAML config parameters to config object
  - Manages user **path** and threads which has to be set before usage, this idea is also taken over from conpy[4]
- Config.yaml:

[4] https://github.com/AaltoImagingLanguage/conpy

- Static config file with relevant parameters over all python files

1.3. MNE Reports

MNE Reports were used to document the results along the pipeline execution. They are conveniently stored inside the reports folder.

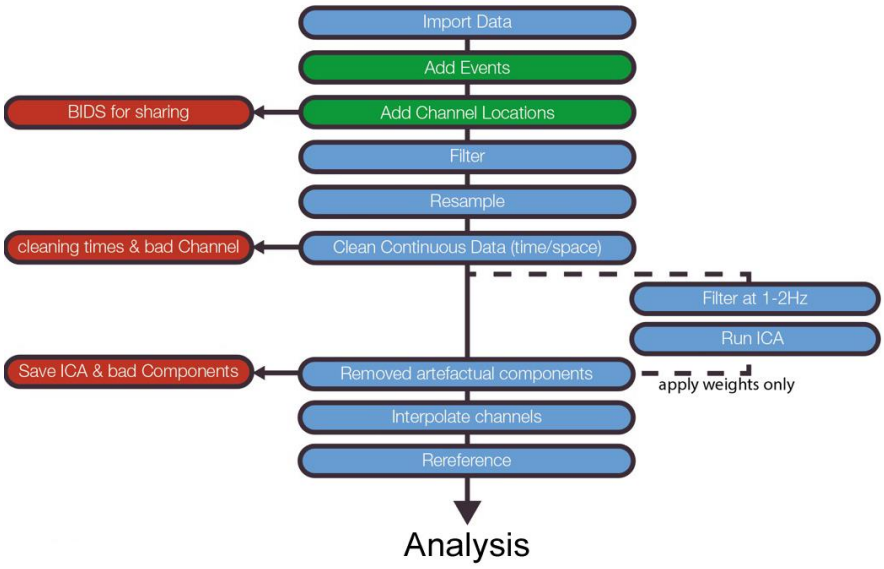


The figure above shows the structure of a subject report. On the left side the different plots can be selected. The top row allows to select for specific report sections.

There exist reports for:

- Single subjects: **e.g. 001-report.html**
- Overall analysis across subjects: **total-report.html**

2. Preprocessing



This chart by Cohen shows the basic preprocessing and cleaning steps for EEG data.
During the preprocessing part this work follows these pipeline procedures step by step. In the next subchapter the steps are explained in more detail including parameter choices.

2.1. Filter

File:	Input artefacts:	Output artefacts:
01_filtering.py	Single subject cleaned data (e.g. 001-filtered-raw.fif)	Single subject data with applied ICA (e.g. 001-cleaned-raw.fif)

First, filtering is performed. It is important to choose an appropriate filter because filters may cause significant distortions/biases to the data. My chosen therefore are based on a set of best practices and guidelines regarding filter design choices.

Filtering the EEG Core data with a bandpass filter provided in MNE. Filter choices:

- **Bandpass-Min: 0.5 Hz**
  Widmann et al. [2] show that high-pass filters with 0.75Hz cut has only minor effects on ERP data. Therefore, I choose to stay below this threshold of 0.75Hz
- **Bandpass-Max: 50 Hz**
  Following recommendations of Widmann et al. [2]: “low-pass filters with cutoff frequencies higher than 40 Hz during ERP analysis are recommended”
- **Filter design: Firwin**
  FIR filters have advantages over IIR filters like [2]:
  - Easier to control
  - Always stable
  - Well-defined passband
  - Etc.

  Additionally, [2] recommend going with a bandpass filter instead of separate low-/high pass filters

2.2. Cleaning

File:	Input artefacts:	Output artefacts:
02_cleaning.py	Single subject cleaned data (e.g. 001-filtered-raw.fif)	Single subject data with applied ICA (e.g. 001-cleaned-raw.fif)

This step performs cleaning of bad segments and bad channels. It can be executed either using precomputed annotations and loading them or in an interactive way which let you choose with MNE plot functions which segments/channels are bad.

- Bad segments:
  - Are saved in the raw file instance as annotations

- Bad channels:
    - Saves them in *raw.info["bads"]* → Which is then used by MNE to drop bad channels in Epochs
    - Interpolates bad channels if there are any (using *raw.interpolate_bads()*)

Cleaning can be performed in several modes:

- "PrecomputeMode": Uses precomputed bad segment annotations and bad channels
- "DialogeMode": Plots the ICA and individual components views directly. This is used for manual cleaning of subjects

2.3. ICA

File:	Input artefacts:	Output artefacts:
03_ica.py	Single subject cleaned data (e.g. 001-cleaned-raw.fif)	Single subject data with applied ICA (e.g. 001-ica-raw.fif)

The Independent component analysis (ICA) allows to split data into individual components which then can be analyzed regarding their effect on the data. The implemented ICA operates the following steps on a **single subject**:

1. Load cleaned raw object from step 02
2. Use ICA specific High pass Filter as described by Cohen in his pipeline and other sources[5].
   This shall remove slow drifts in the data before computing the ICA.
   Applied High pass with Cutoff-Frequency 1Hz.
3. Fit the ICA and remove bad components to the initial raw data object
4. Save the new raw object

The implemented ICA has several modes to choose from:

- "PrecomputeMode": Uses precomputed provided ICA from EEG lab
- "DialogeMode": Plots the ICA and individual components views directly. This is used for manual cleaning of subjects

As ICA technique it was chosen to use *picard* because according to MNE docs it is the fastest converging ICA technique implemented.[6] Also a random_state was selected to be reproducible when performing manual cleaning again to get the same components.

Bad ICA components are hard coded for the given subjects inside the config file.

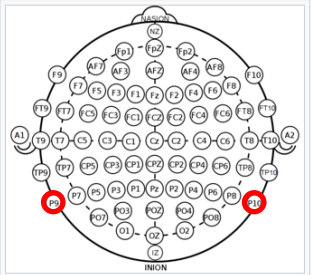
2.4. Rereference

File:	Input artefacts:	Output artefacts:
04_rereference.py	Single subject cleaned data (e.g. 001-ica-raw.fif)	Single subject data with applied ICA (e.g. 001-refererenced-raw.fif)

[5] https://mne.tools/dev/auto_tutorials/preprocessing/plot_40_artifact_correction_ica.html
[6] https://mne.tools/dev/generated/mne.preprocessing.ICA.html

Selects the reference electrodes for the signal. The goal is to provide a clean signal free reference point (silence point) which serves as reference voltage for the signal amplitudes. Referencing keeps relation between the data.

Guidance for selecting a good reference was taken from several sources in the literature e.g. ERP Core Paper [3] and several Tutorials[7] [8]



Elektroden-Anordnung im 10-20 EEG-System

https://de.wikipedia.org/wiki/10-20-System

It was found that there are several possibilities to choose the reference. Modern approaches recommend the Average of the electrodes as Reference, whereas the ERP Core Paper uses the of the mastoid sites (P9, P10). Besides these two approaches also the REST (Reference Electrode Standardization Techniques) can be used. While experimenting with referencing conditions it was decided to stay close to the ERP Core paper by using the local adjacent to the mastoids (average P9/P10) as reference. But the pipeline also still supports average referencing via config.

The reference script operates as follows:

1. Load data from previous step (ICA)
2. Perform the referencing
   Here the script allows or several possibilities via config file. One can also choose the average mode for referencing which then also uses MNEs projectors to store the reference independent of the data (raw data stays consistent)
3. Generate referencing plots

2.5. Analyze

File:	Input artefacts:	Output artefacts:
05_analyse.py	Single subject referenced data (e.g. 001-rereferenced-raw.fif)	Single subject epoch data(e.g. 001-coded-epochs-epo.fif)

The analyses step calculates epochs for the given subject and plots several images as visual checks to judge about subject quality, individual evoked difference between conditions of the subject, topography plots.

It operates in the following order:

1. Read raw file from previous step(referencing)
2. Get epochs from raw data (utils.py)
   a. **Epochs are encoded with their conditions:**

```
"11": "Stimulus - block target A, trial stimulus A",
"21": "Stimulus - block target B, trial stimulus A",
"31": "Stimulus - block target C, trial stimulus A",
"41": "Stimulus - block target D, trial stimulus A",
"51": "Stimulus - block target E, trial stimulus A",

"12": "Stimulus - block target A, trial stimulus B",
"22": "Stimulus - block target B, trial stimulus B",
"32": "Stimulus - block target C, trial stimulus B",
"42": "Stimulus - block target D, trial stimulus B",
"52": "Stimulus - block target E, trial stimulus B",
```

P3 experiment coding

[7] https://mne.tools/stable/auto_tutorials/preprocessing/plot_55_setting_eeg_reference.html
[8] https://www.fieldtriptoolbox.org/faq/why_should_i_use_an_average_reference_for_eeg_source_reconstruction

For the P3 the event codes for the stimuli from 11 up to 55.
Responses are codes 201/202. For each of these trial event codes the actual condition is replaced. This means the oddball in that given trial (11,22,33,44,55) is encoded as **"cond1"** and the other stimuli codes with **"cond2".** The naming is also changeable via config file. They are kept very general which allows the pipeline to be **task independent.** The names can also be changed to arbitrary names like "rare" and "frequent" and the event codes are also fully customizable.

b. Remove epochs with wrong responses

Epochs where the response code is 202 are calculated and their stimulus and response segments are deleted from the epoch. Deletion is achieved by searching the event codes of all epochs for the given bad response code and then remove this epoch and the epoch before where the stimulus was.
This allows to clean parts where the experiment subject answered wrongly to what was visible on the screen. These data units would introduce unwanted noise to the analysis as they do not hold the information that shall be analyzed . When the subject did not correctly process what was shown on the screen one cannot argue the associated effects to this behavior. Therefore, it gets removed before further analysis. There are also several checks to verify that this removal worked correctly, which is checked programmatically and can be seen by the user as well in the report event id plots.

c. Baseline correction

After inspecting the data for several subjects, it was decided to go with a baseline ranging from signal start to 0 sec. This means that in this given time window the mean of the signal is calculated and then subtracted for all epochs of a single subject and channel individually. To apply this baseline correction the MNE default settings **(None, 0)** are used. [9]
Inspecting the baseline visually is also important therefore most of the plots include the baseline before 0s

3. Generate evoked difference wave between conditions
4. Plot several visual checks
5. Save epochs

2.6. Grand Average

File:	Input artefacts:	Output artefacts:
06_grandAverage.py	Single subject referenced data (e.g. 001-rereferenced-raw.fif)	Single subject epoch data(e.g. 001-coded-epochs-epo.fif)

[9] https://mne.tools/stable/generated/mne.Epochs.html

The Grand Average script operates over subjects and computes the evoked difference between the subject conditions. Procedure is as follows:

1. Iterate over subject epochs and load each subject saved epoch object
2. For every subject calculate the evokeds for both conditions and the difference wave
3. In each subject search in the provided time window search for a voltage peak
4. After iterating over subjects: Create average of subject individual evokes & difference wave average
5. Create plots and print voltage peak distribution of subjects
6. Calculate statistics: perform one-sided 1-sample T-test between subject conditions that is comparing all peak values of the difference waves from subjects to zero.
   More details on the statistics can be found in (Ch. 3.4 Peak analysis)

2.7. Time-frequency

File:	Input artefacts:	Output artefacts:
07_tfAnalysis_01.py	Subject epoch data(e.g. 001-coded-epochs-epo.fif)	Subject time-frequencies (e.g. 001-tf-power.pickel)
08_tfAnalysis_02.py	Subject time-frequencies (e.g. 001-tf-power.pickel)	-

Further details are explained in (Ch. 3.5 Time-Frequency analysis)

2.8. Decoding-analysis

File:	Input artefacts:	Output artefacts:
09_decoding_01.py	Subject epoch data(e.g. 001-coded-epochs-epo.fif)	Subject decoding scores (e.g. 001-decode-peaks.pickel)
10_decoding_02.py	Multiple Subject decoding scores (e.g. 001-decode-peaks.pickel)	-

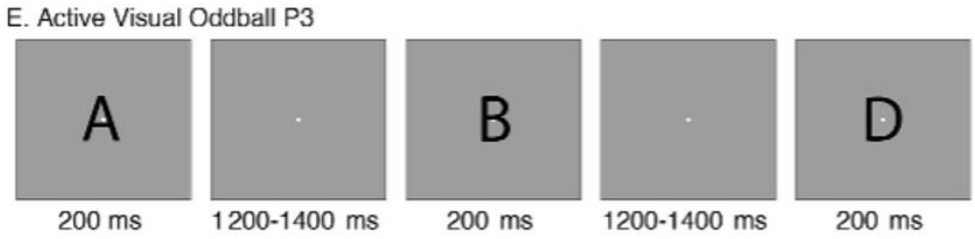
Further details are explained in (Ch. 3.6 Decoding analysis)

3. P300/P3 Analysis: Visual oddball experiment

3.1. Task description

The oddball experiment is used to elicit the P3 in experiments. The task consists of a set of letters (A, B, C, D, E) which are presented to the subject in arbitrary order (p=0.2) . In each trial one of the letters was declared to be the oddball. Participants then watched the letters appear and indicated if it was the target or not. [3]

The following figure from [3] shows how the experiment looks like:



3.2. ERP Core dataset

The ERP Core dataset is a collection of freely available scripts/data/analyses of several ERPs including the P3/P300. The datasets from the ERP Core are used for this work. [3]

Experiment data in the ERP Core was conducted using 30 scalp electrodes. The ERP Core data has some additional signal processing for the data, this includes:

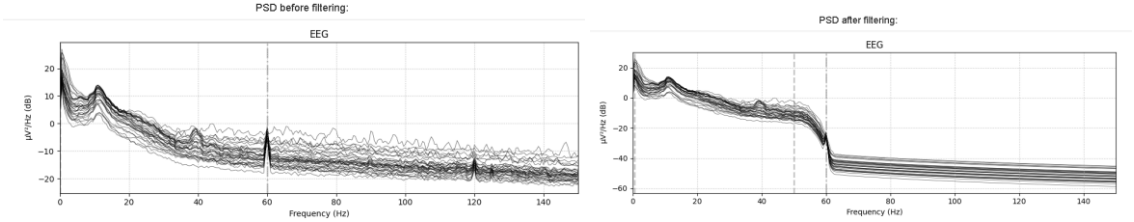
- **Monitor delay:** Data from the ERP Core are already shifted to for account for LCD monitor delay, so this must not be done in the project pipeline
- **Resampling:** Subjects are already resampled to 256Hz to improve data processing times

3.3. Manual preprocessing subjects

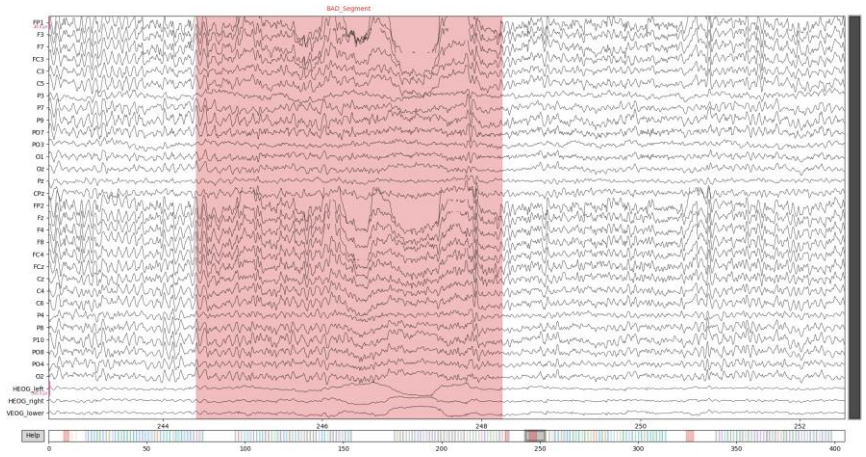
After data inspection it was decided to clean subjects **002, 009 and 040**. Especially subject 009 seems very noisy also with possible damaged channels. Therefore, perform analysis further.
Manual processing can be done&reproduced by setting the ***isDialogeMode: True*** in the config and **disabling the PrecomputeMode.**

3.3.1 Subject 002

1. Filtering
   For filtering the standard choices for all subjects where used. The reasons for these are explained in Pipeline step Filtering. The following plots show results for Subject 002.



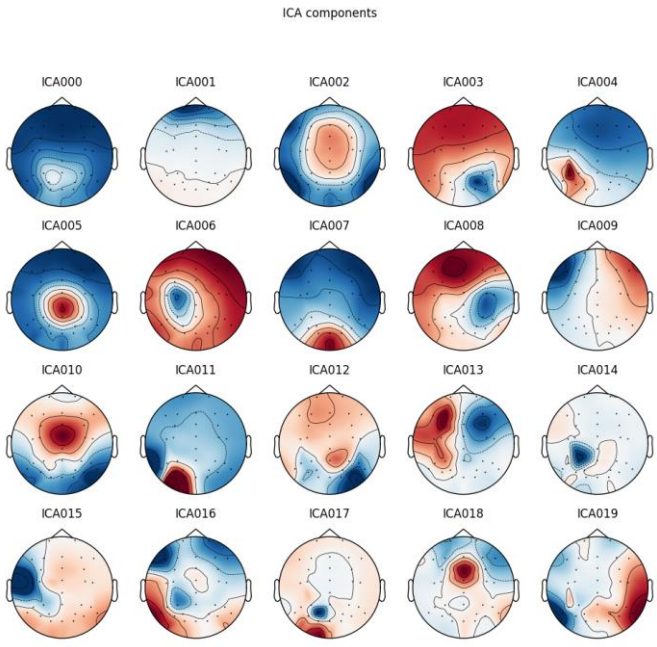
2. Cleaning



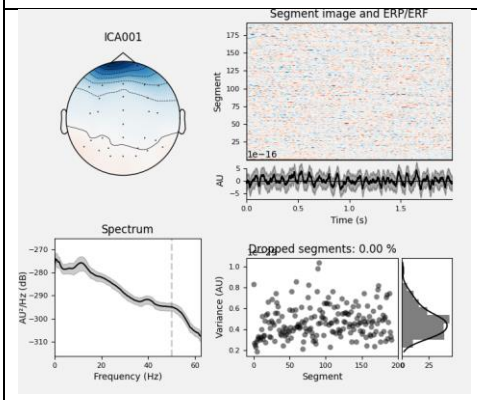
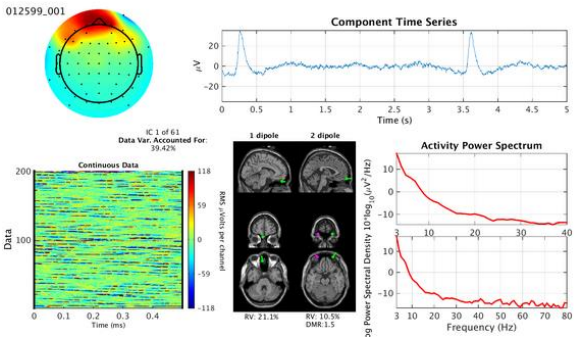
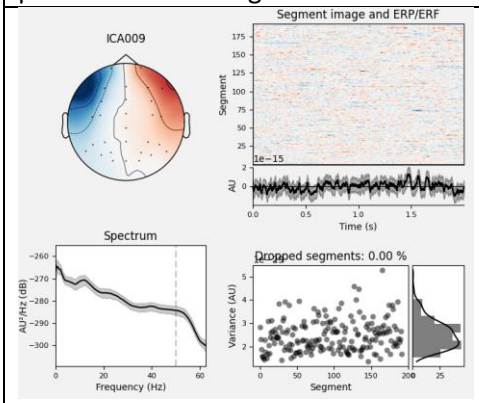
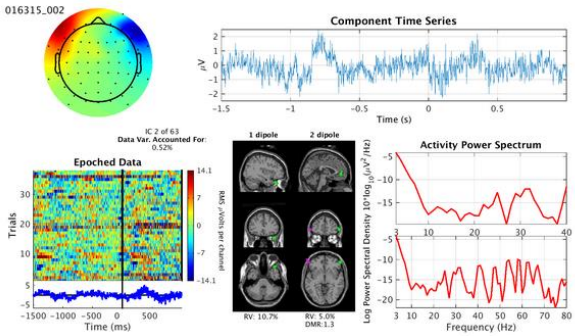
- Bad segments parts are marked with MNEs interactive annotation mode
- Full cleaning can be seen in the annotations file for subject 002 (002-manual_clean-annotations.txt)
- For subject 002 no bad channels were found during manual inspection

3. ICA

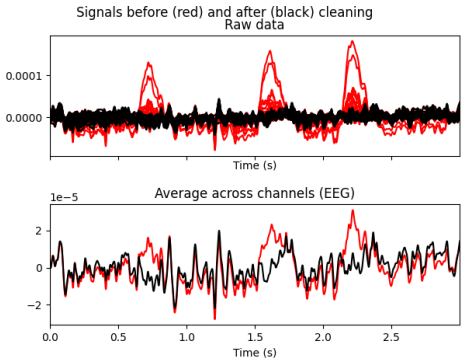
Overview ICA components subject 002:



- Next step: Individually watch at each component and judge if it is component to be removed like eye artefacts, muscle artefacts etc.

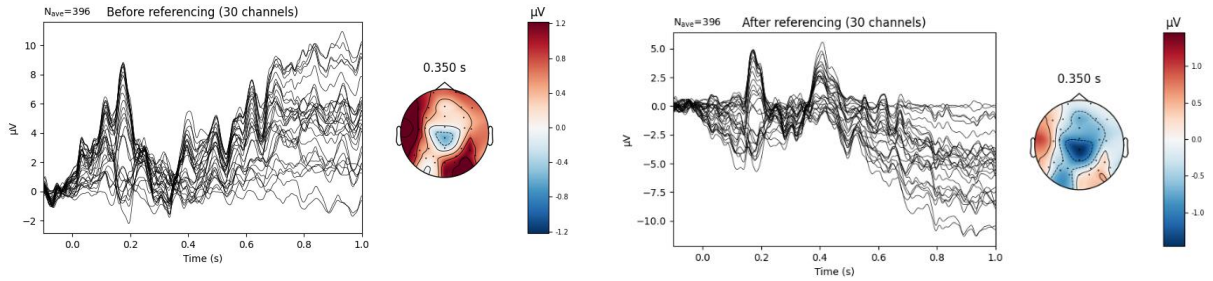
Subject ICA Component	Reference of ICLabel Tutorial (https://labeling.ucsd.edu/tutorial/labels)
 Component 001 is probably an eye artefact given the topography plot the effect is present at eye electrodes. High power <5Hz and in general	
 Eye artefact: Probably Horizontal eye movement given the topography plot	 This components captures the effects of horizontal eye movement, although some high frequency power in included from some other source.

After applying the ICA and removing components: **ICA 001 & ICA 009**



In the overlay plot of ICA data before and after it becomes clear how the bad components influenced the signal ad one can directly see which improvements were made.

4. Re-Referencing:

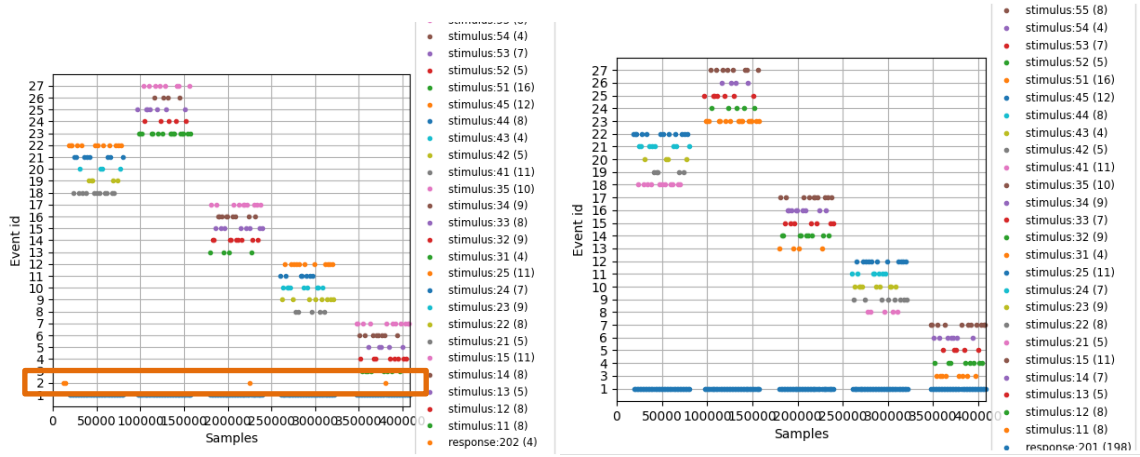


One can see the drastic effects on signal representations across channels using re-referencing. This shows again the importance of selecting and applying re-references to the signal.

5. Further subject individual analysis:

Additionally, to the preprocessing steps for the individual subjects, many more analysis plots are generated. These will be only displayed here for subject 002, to keep the report smaller but they are for every subject available in the corresponding subject **HTML report** (e.g. 002-report.html).

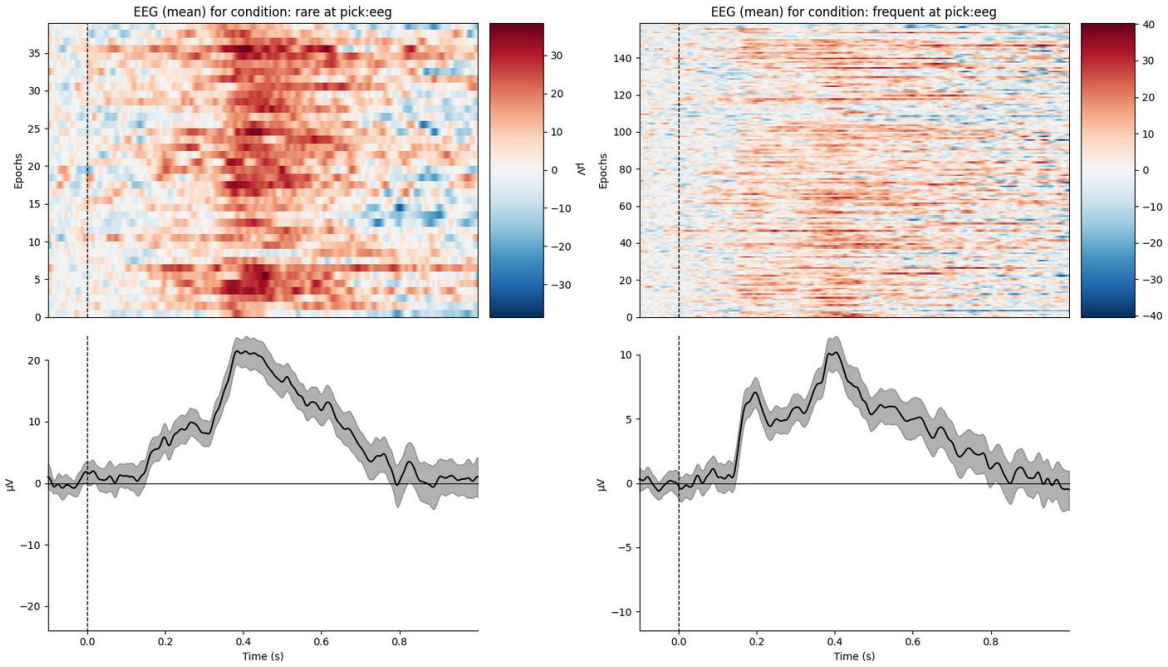
Bad response removal:



This plot visualizes the event times. The 5 trials for each subject become very visible. Additionally, this plot is used to check the removal of bad responses as explained earlier. One can see that responses with the bad index (202 marked with orange box) are removed. Left to Right figure here shows before/after cleaning.

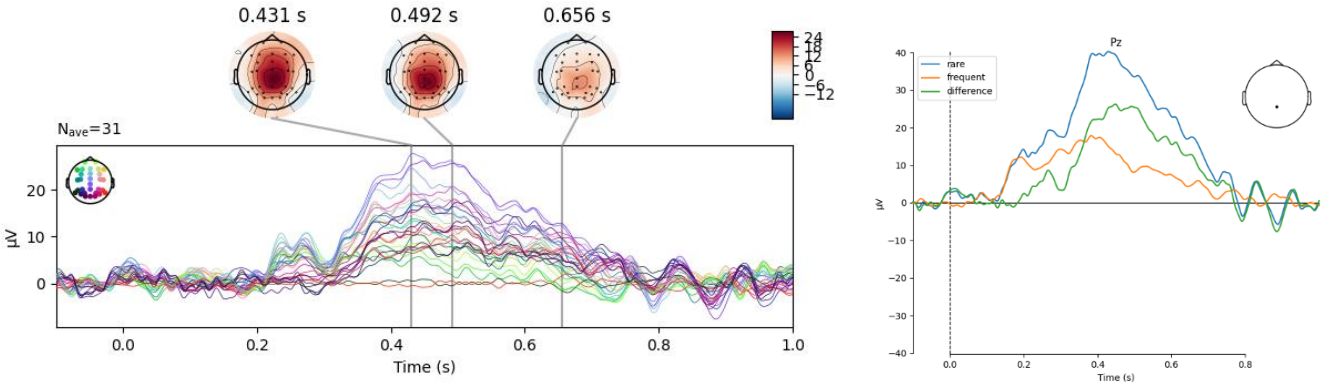
Epoch inspection:

These figures visualize the epochs separated by conditions, their voltage over epochs and the corresponding evoked. The visualization currently shows epochs over all EEG channels. This can be changed via config. One can see that there are much less rare then frequent epochs. This will become later important in Decoding.

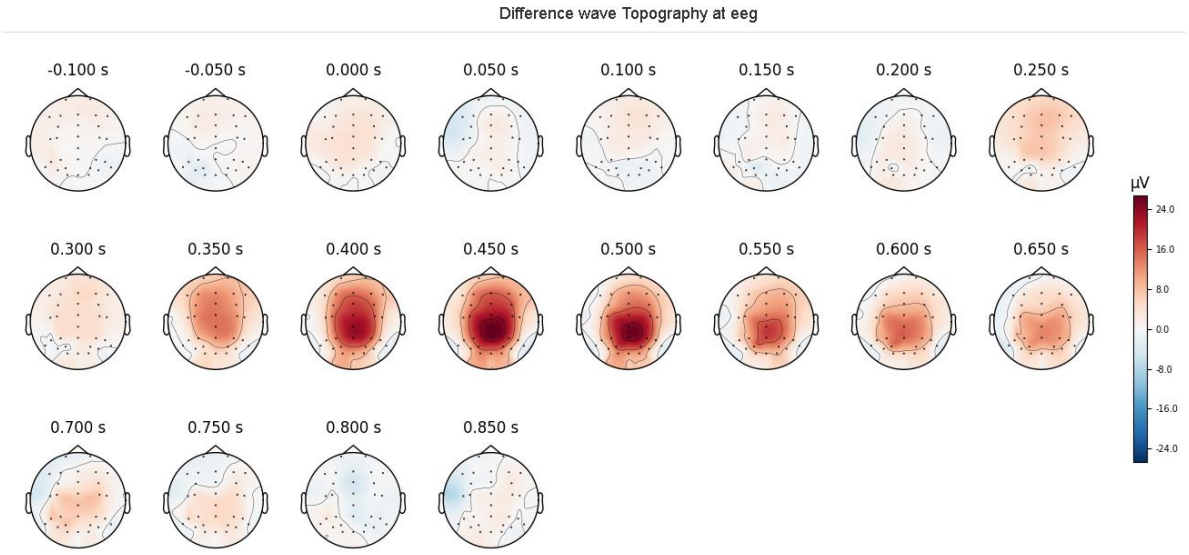


Subject individual evokeds and ERP analysis:

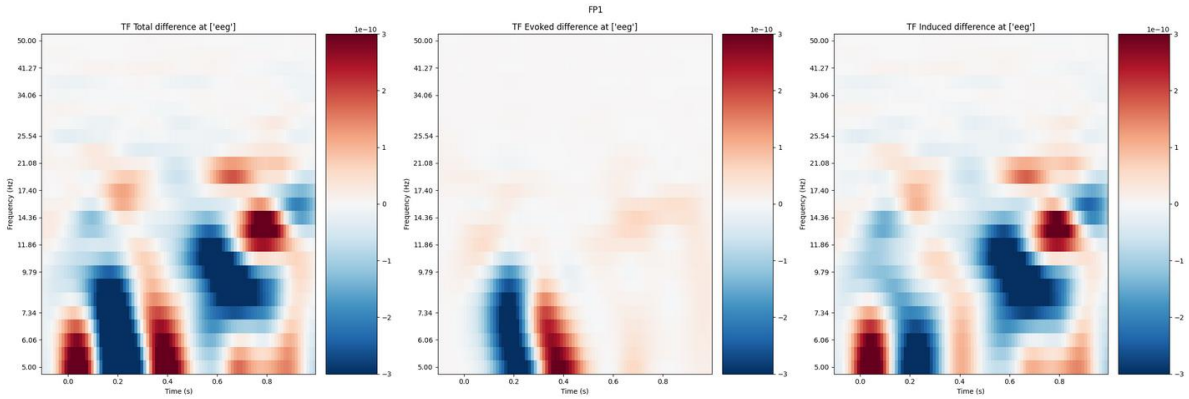
Getting a better understanding of the experiment data and the subjects is a key part of the analysis. Therefore, several different plots for the subject individual ERPs are generated.
Besides the single electrode plot one can also see all channels overlayed. From the color coding for the electrodes it is visible that the ERP is most dominant in the mid brain electrodes.



Further insides can be obtained when looking at the topography plot, which is shown for the whole timespan across all EEG channels. Also baseline period is visualized to ensure no unwanted biases are introduced to the data analysis.



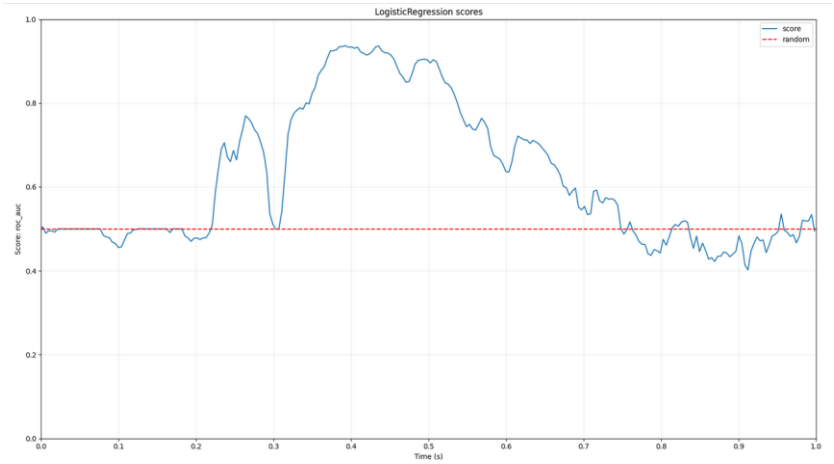
Subject individual time-frequency analysis:



For each subject the time-frequency analysis with evoked & induces time-frequency representations are visualized.

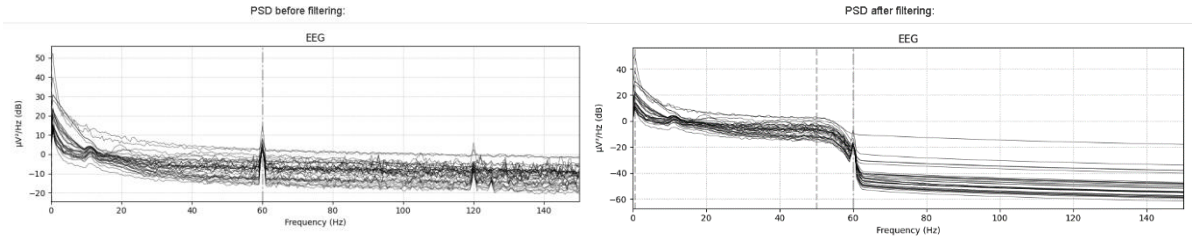
Subject individual decoding analysis:

Also decoding is performed for each subject individually first, which can easily be inspected inside the report.

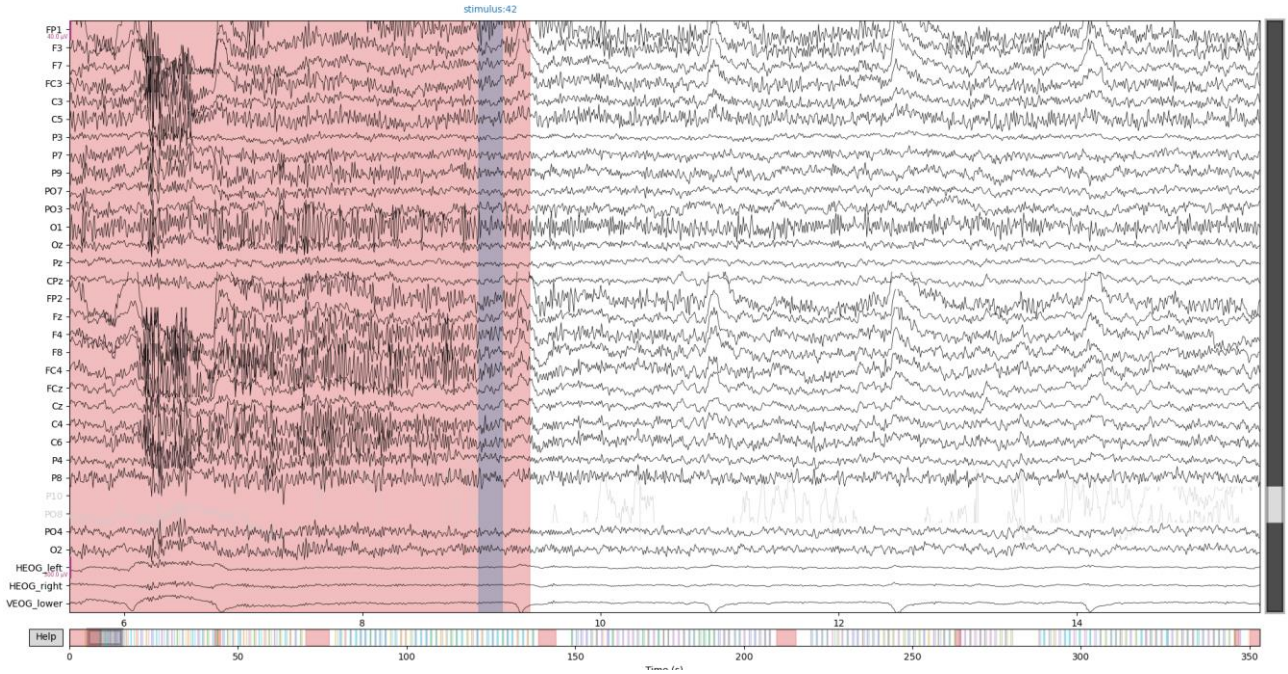


3.3.2. Subject 009

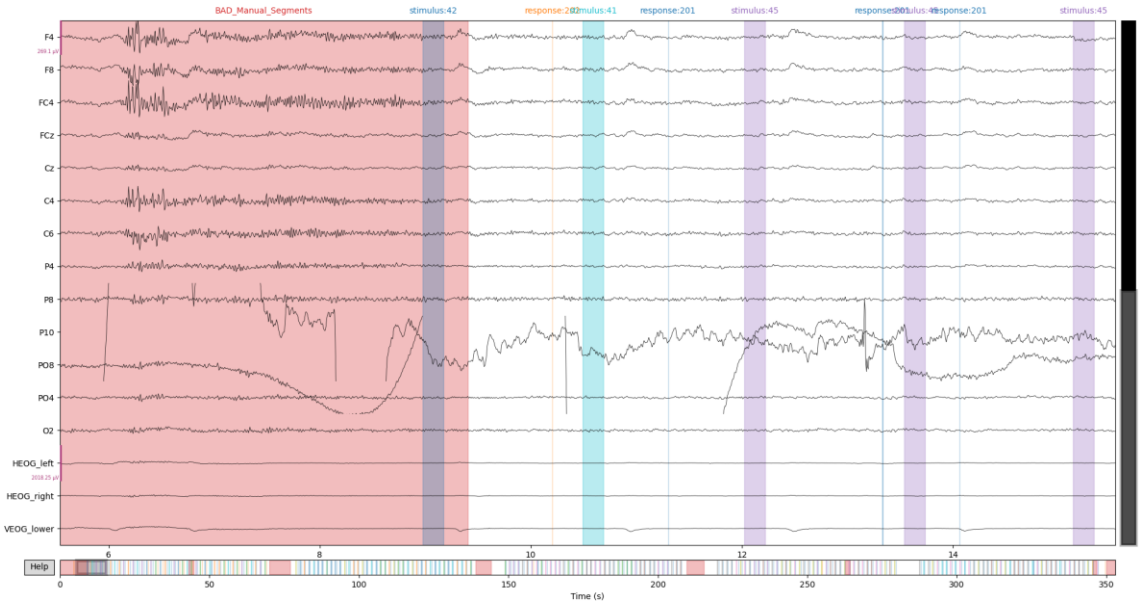
1. Filter



2. Cleaning

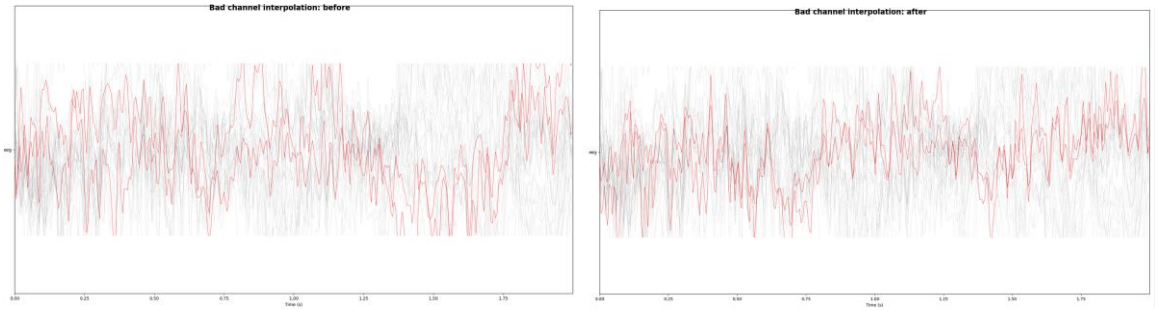


Subject 09 has some very noisy segments which are carefully cleaned by marking the segments in MNE. Segments cleaning → also in file available



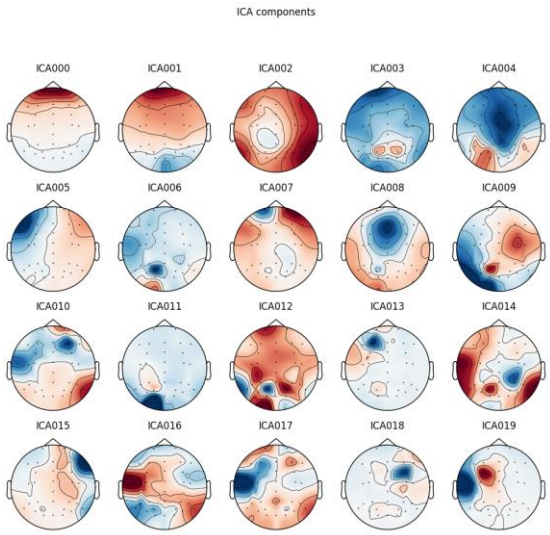
When decreasing the signal scaling and zooming into channels it becomes clearly visible that Channels **PO8& P10** are bad channels for this subject. These are marked also in the interactive plot, by clicking the channel name (greyed out).

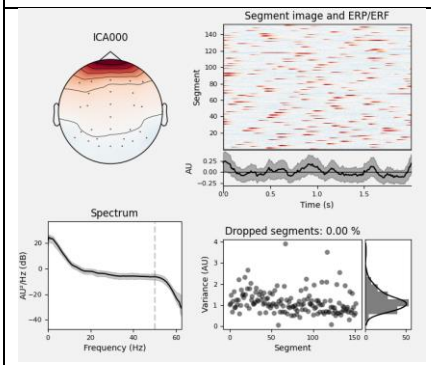
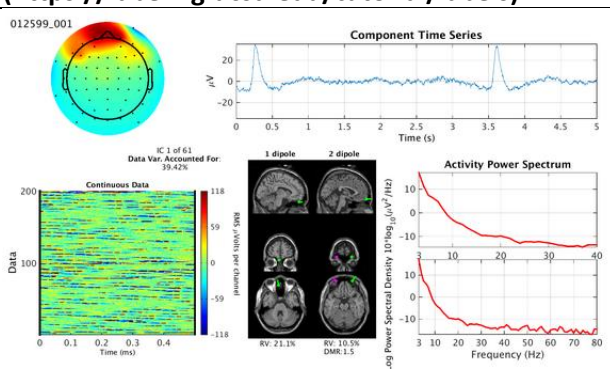
Bad channel interpolation plots:

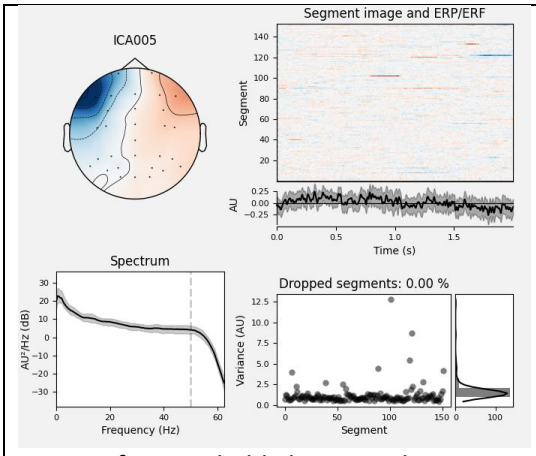


The figure above shows difference before and after interpolation. It is a bit difficult to see and judge the interpolation, but in general the bad channels (red) move close to the other channels signals by appyling the interpolation of MNE.

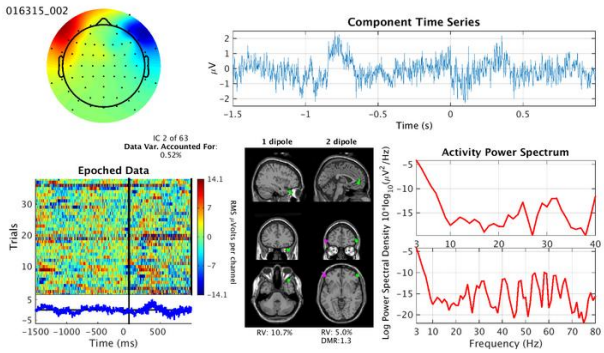
3. ICA



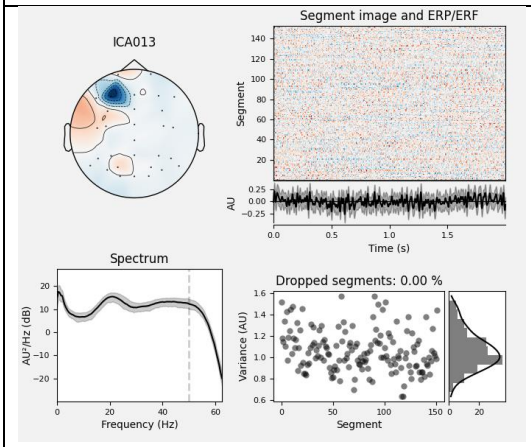
Subject ICA Component	**Reference of ICLabel Tutorial (https://labeling.ucsd.edu/tutorial/labels)**
 Eye artefact	



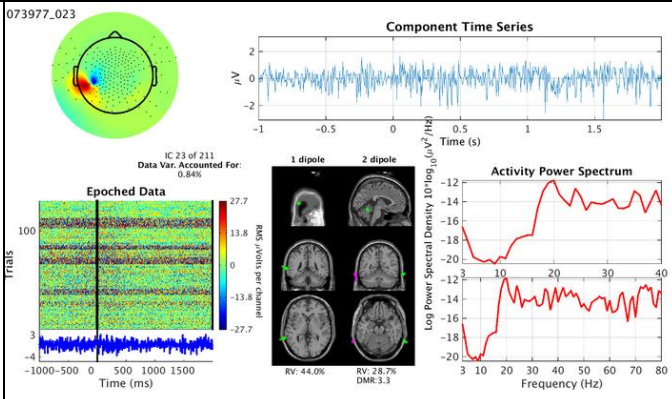
Eye artefact, probably horizontal eye movement, decision based on frequency at location of artefacts



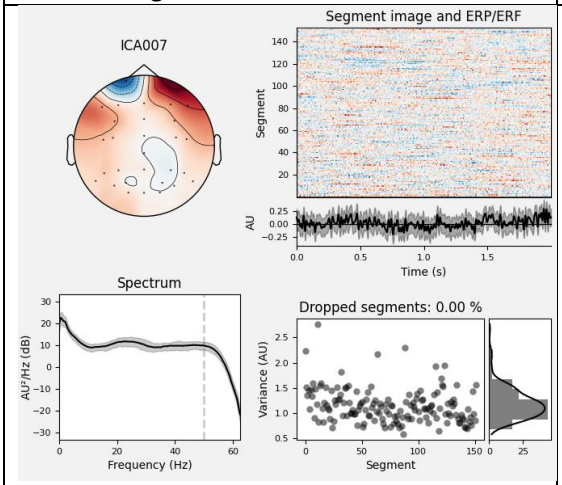
Horizontal eye movement example



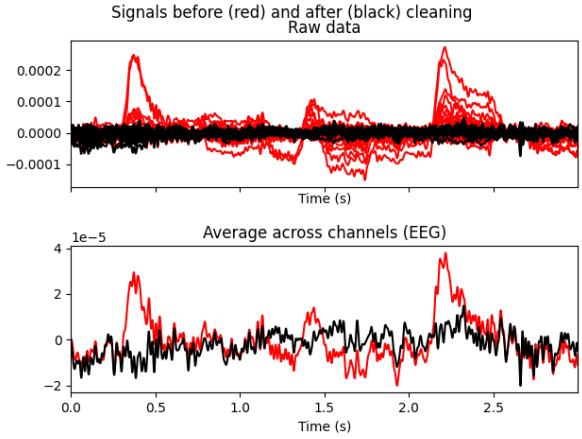
Probably Muscle artefact, Frequency increases again at 20Hz and above



Comparison of muscle artefact example

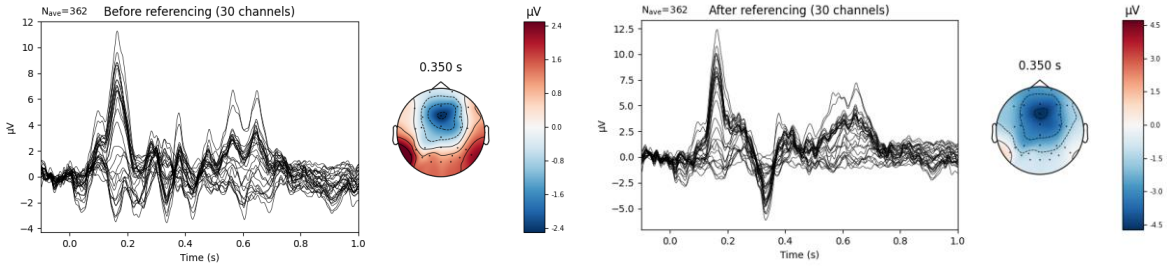


Eye artefact, based on artefact location



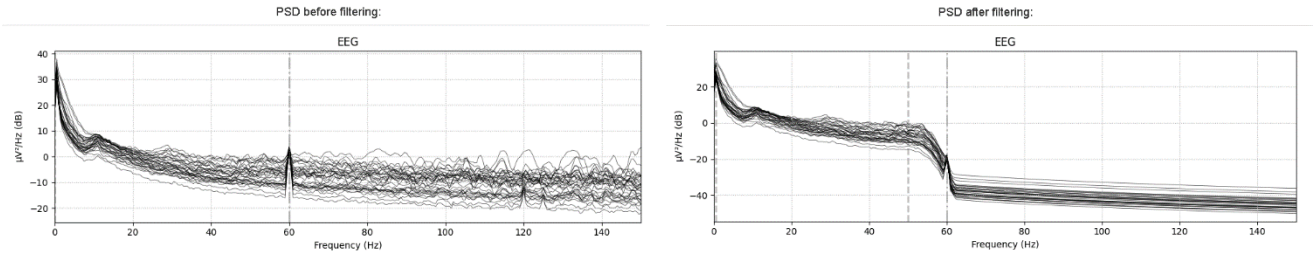
Plot shows overlay of bad components (red) over other components (black). Visually the signal now looks much cleaner as noisy signals that diverge largely from the others seem removed.

4. Re-referencing

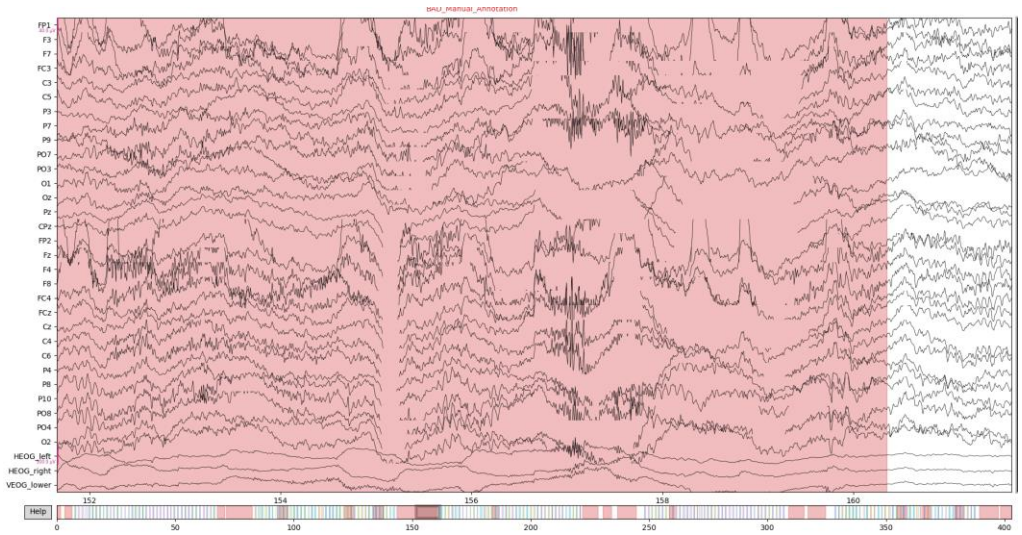


3.3.3. Subject 040:

1. Filter

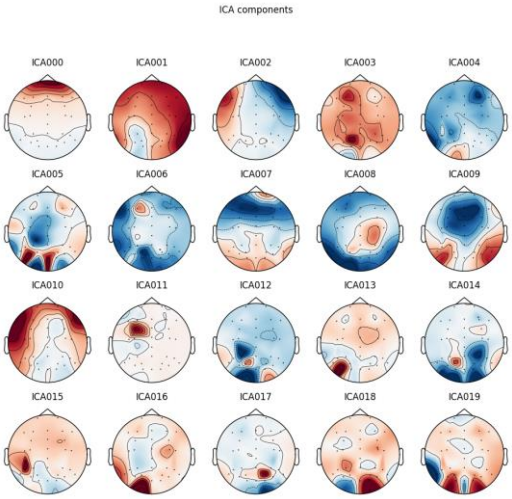


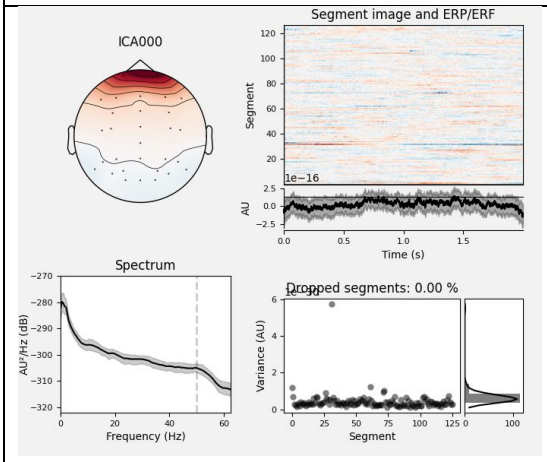
2. Clean segments & channels

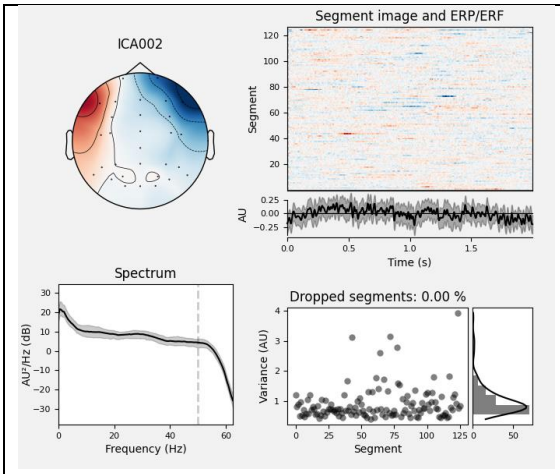


- From inspecting the data **no bad channels** were found

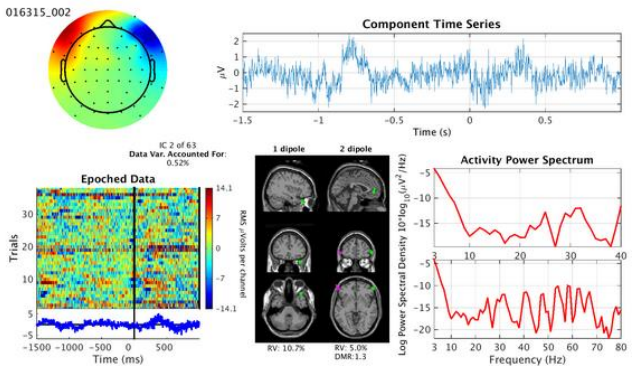
3. ICA



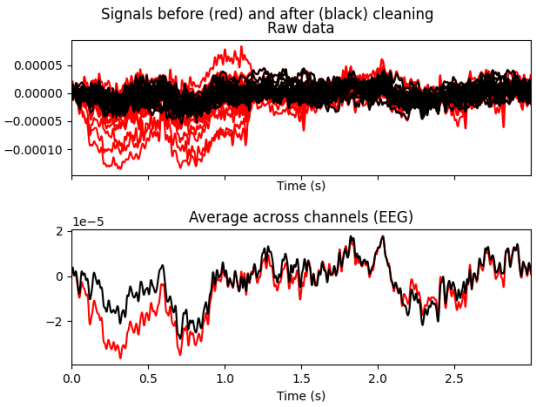
Subject ICA Component	**Reference of ICLabel Tutorial (https://labeling.ucsd.edu/tutorial/labels)**
 Eye artefact	



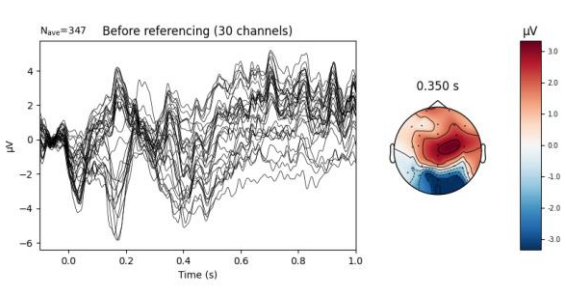
Eye artefact, Probably horizontal eye movement

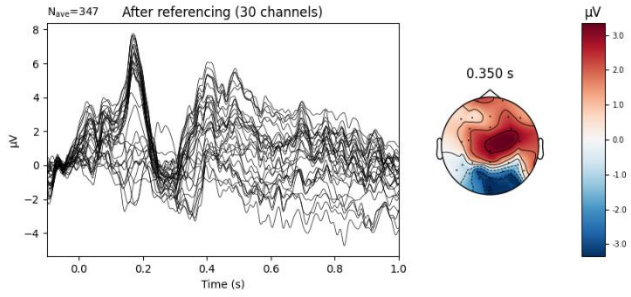


Before/After ICA overlay:



4. Re-referencing

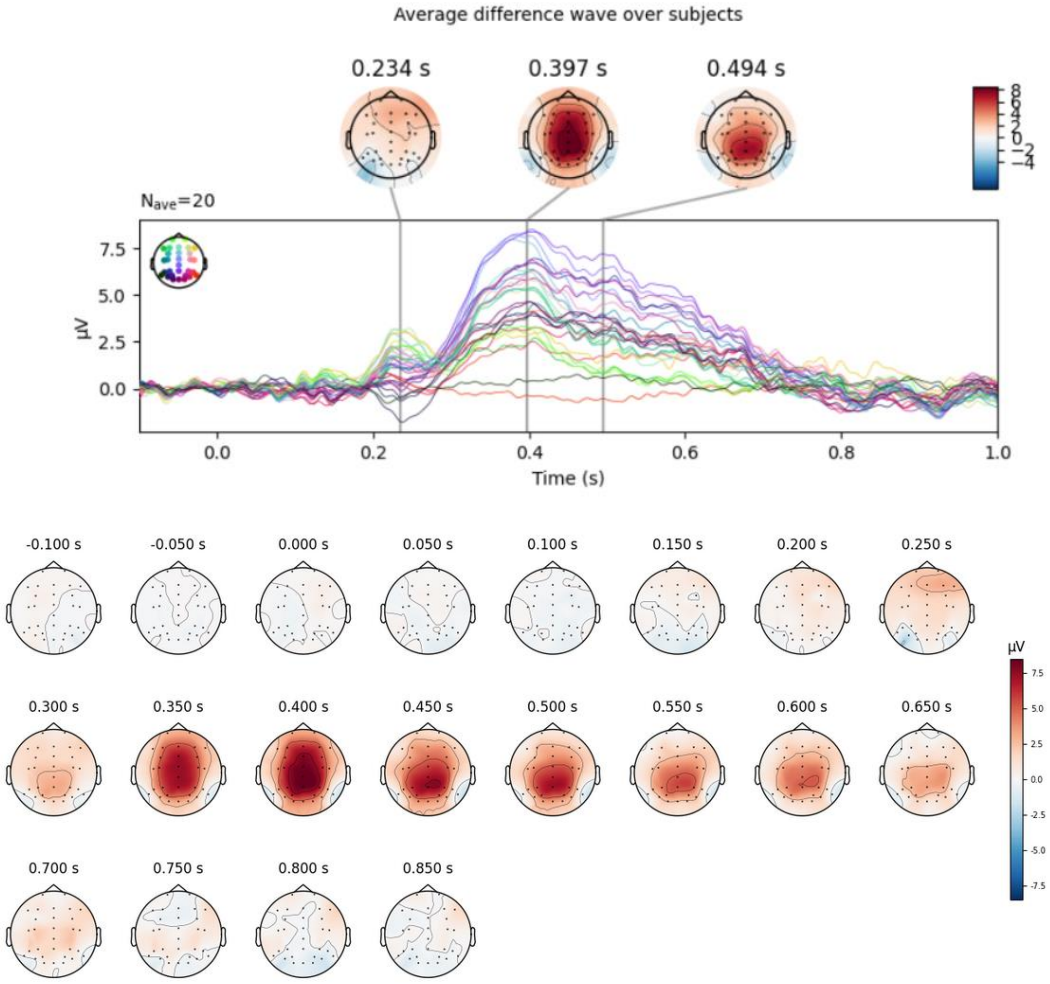




3.4. ERP Analysis

After processing the individual subjects now, the analysis of the ERP is done. For this the goal is to generalize the ERP that was found in single subjects by comparing the differences over all the subjects and applying statistical methods.

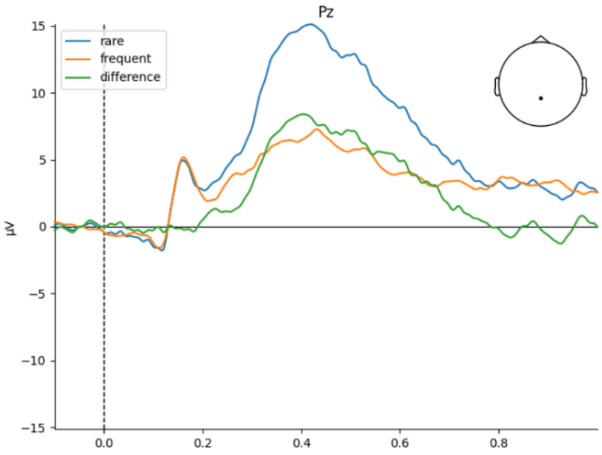
First for each subject the difference of the condition evokeds is calculated. The conditions to evaluate in case of the P3 Tasks are **Rare** (Oddball) and **Frequent(**other latter's in trial). The evoked then is just calculated by taking the average over epochs of each condition individually. Now an average signal over trials exists for each condition. These are the subtracted to get a difference wave for each subject. The reason to do this is to find the difference in these conditions. This becomes also visible when looking at the grand average plot in the following figure.



1 Topography plots for ERP difference wave grand average analysis

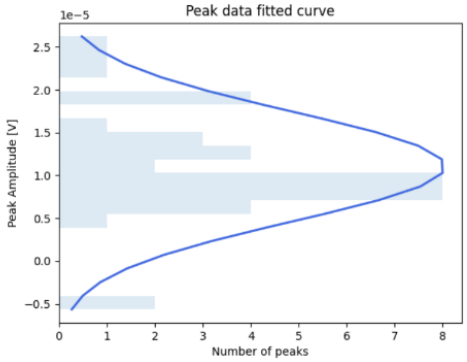
In the topography plot one can already see that there are strong artefacts starting at ~300ms. The voltage is especially high in the brain midline around electrodes Pz, Cz, CPz.

For further analysis now the Pz electrode is chosen. Pz is chosen for further analysis of this effect as it was visible in the topology plot that the effect seems to be strong there. Also in literature the Pz electrode is mostly used for P3 tasks as it is reported in the ERP Core paper [3]. But one could also go with nearby electrodes like Cz, CPz etc.



P3 Grand average of ERP at electrode Pz

Here one can see the voltage curves depending on the condition.
The next step is to analyze this peak statistically.



For the statistical analysis peaks are extracted from the difference wave (green wave in Grand average) per individual subject.
In the histogram the extracted peaks and their distribution can be observed. On first qualitative inspection they seem to are somehow normally distributed, but one would have to obtain much more data to make this more visible. For the peak extraction a time window was specified to narrow down the expected effect window to 300ms – 600ms. According to the topology plot and the literature for this topic this seems to be a valid choice for peak detection.

To test the effect a two-sided T-test using *scipy.stats.ttest_1samp* is applied, which calculates for H0 that the expected value of independent observations is equal to population mean. [10]

The statistical Hypothesis are:
H0: No difference between the conditions rare&frequent → Difference wave = 0
H1: There exists a difference between the condition → Difference wave ≠ 0

The results on our data with a significance level alpha of 5% are:

T-test: p-value 2.8779381295991733e-13 < 0.05 alpha

According to the T-Test and p-value with alpha =5% the Null-Hypothesis is rejected and the alternative H1 accepted. Probably there is a statistically significant effect between the conditions given the data evaluated at electrode “Pz” and peaks in time window 300ms-600ms.

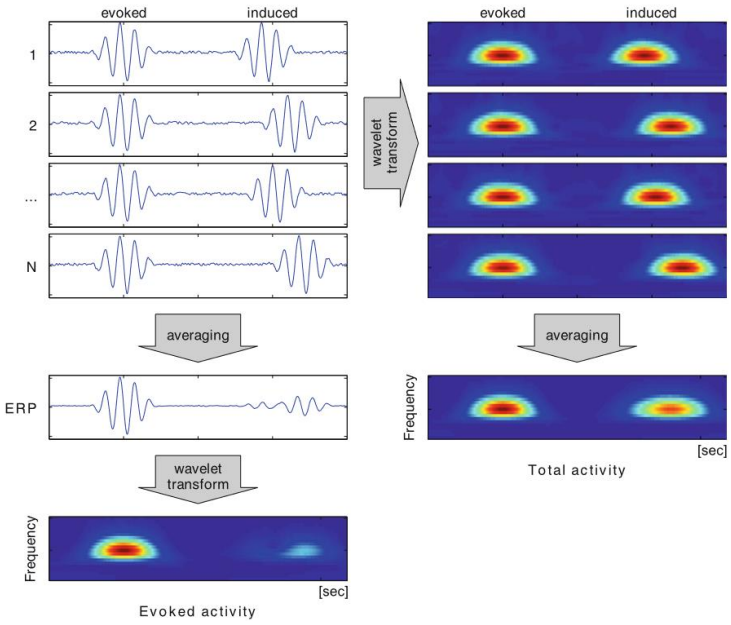
[10] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_1samp.html

3.5. Time-Frequency analysis

Calculate an induced time-frequency analysis of the main experimental contrast *RQ: What oscillations underlay our effect of interest?*

With time frequency analysis a signal can be further inspected beyond the time domain, by including the frequency spectrum to get a better understanding of the signal and possibly find underlaying oscillations. The given ERP data shall now be analyzed in the time frequency domain.

One big advantage is in the time-frequency domain is that there the **non-phase locked** is not canceled out by computing the average from epochs over subjects and trials.
This figure by Herrmann et al. [4] shows quite well, that instead of averaging over trials now a transformation into the time-frequency domain shall be performed to keep the induced effect.



The time-frequency analysis is implemented the following way:

Compute subject individual time-frequency data:

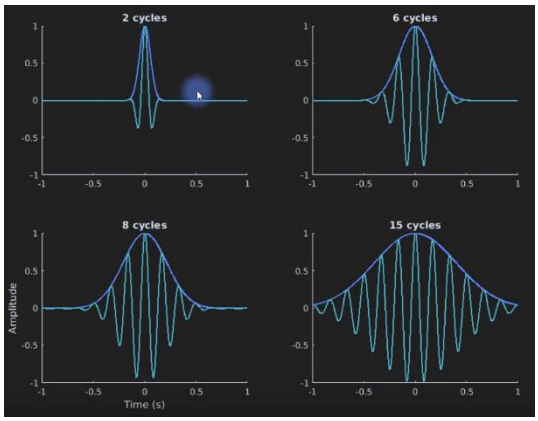
1. <u>Read subject epochs and compute the induced</u>
   The induced signals can be obtained by subtracting the average over the epoch data. By computing the average, the induced is from the data eliminated and get the evoked. After subtracting the evoked from the total one obtains the Induced.

   $$Induced: NonPhaseLocked = Total - phaseLocked$$

   The induced is computed for both conditions separately, because later the difference of the experimental contrast, which is the difference in conditions, shall be analyzed

2. For comparison and visual inspection not only induced but also total and evoked power are calculated

3. Apply Morlet Wavelets to compute time frequency representation (TFR) of our evoked, total and, induced effects. When calculating the Wavelets, it is important to choose the correct frequencies and cycles for the wavelet. The number of cycles defines the width of the wavelet, which corresponds to the precision one can obtain in the time domain. Narrow wavelets are precise in time but are wide in the frequency domain and therefore imprecise in the frequency range and vice-versa.
   This work uses not a fixed number of cycles or frames, but rather chose the cycles frequency dependent. It was found after comparing several parameter choices, that cycles = frequency/2 works well for the given ERP data



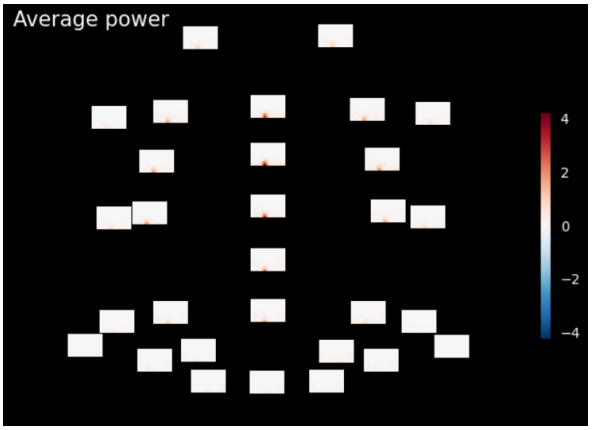
2 Wavelet for different cycles, Cohen's Youtube course: https://www.youtube.com/playlist?list=PLn0OLiymPak2BYu--bR0ADNBJsC4kuRWs

4. Calculate differences in the TFR between the conditions to find underlying effects, especially the induced effect which was not discovered by looking at the ERP evoked.
5. Save subject data for further analysis steps

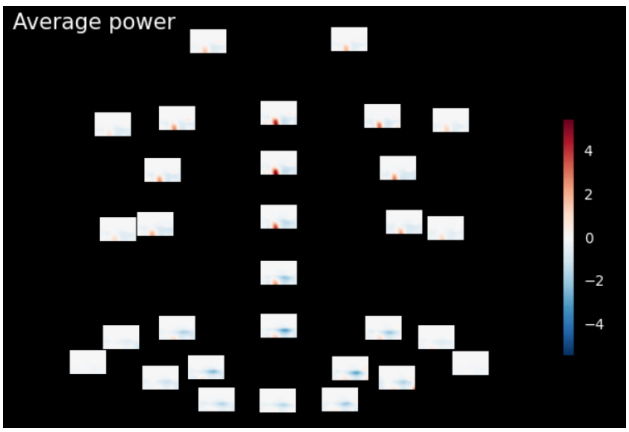
Compute time frequency analysis across subjects and compute statistics:

1. Load subject individual data
2. Combine data over subjects to get average TFR of the experimental contrast
3. Plot visualizations
4. Calculate which effects are significant using a cluster permutation test:
   Goal is to find statistical differences between two groups which are our two conditions rare and frequent. This is achieved by passing multiple observations for each group into the test and compare if they are statistically different. The test is implemented not by using the two conditions but rather by taking the induced differences and compare them against zeros.

For our P3 experiment data the following results are obtained. First having a look at the topology over electrodes of the average induced and evoked power:



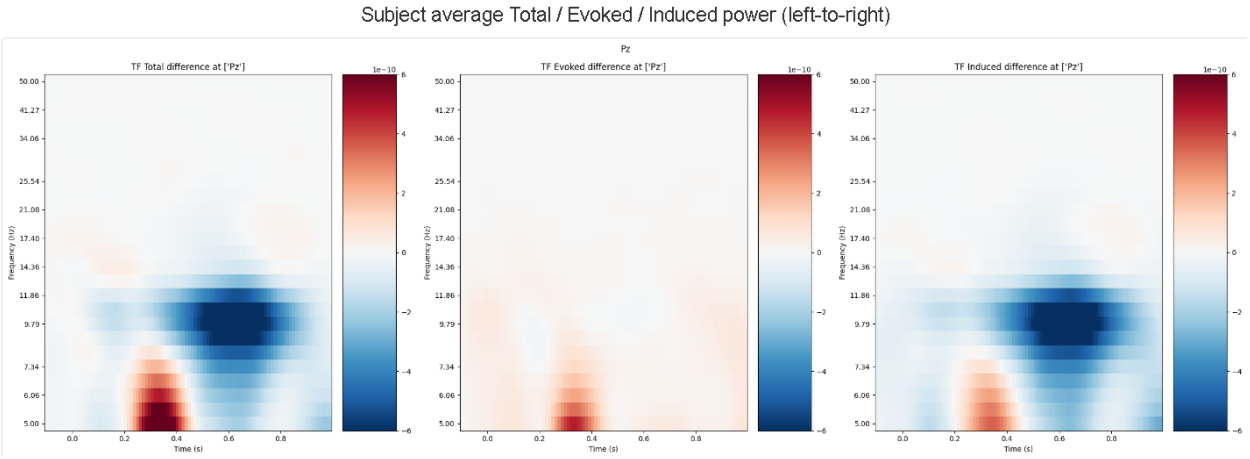
4 Average Evoked



3 Average Induced

The evoked effect that was already statistically tested seems to be in the brain midline as expected. The induced non-phase locked activity seems to be especially large in the frontal area of the brain, also there is obvious negative power in the right back side of the brain.

For further analysis the time frequency plot at our electrode Pz, which was already picked for the ERP analysis is chosen, as the effect is present there and underlying oscillations of this effect shall be found.

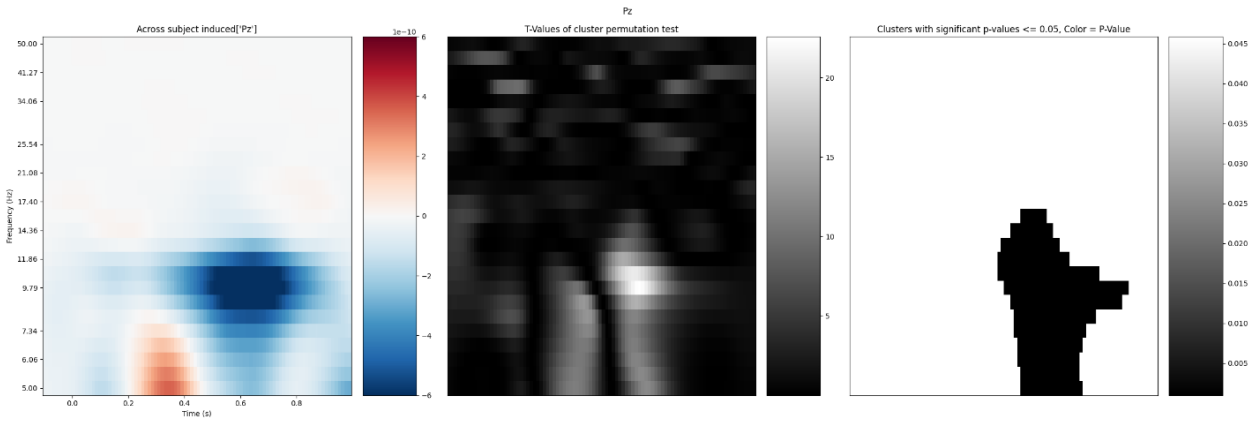


Here it is visible again that the evoked is present in at ~300-400ms, where it was also in the grand average and where it got tested already. Additionally, an induced non-phase locked artefact is visible. Around ~300-400ms right where our evoked is are also induced lower frequencies. Which means the ERP has several non-phase locked signals that overlay the signal in that given time. There also is a quite negative power artefact of the induced at around ~600ms in higher frequencies.

These artefacts shall be tested if they are significant induced effects. To evaluate significance a Permutation cluster test (mne.stats.permutation_cluster_test)[11] is performed. The goal here is to find to find time points differing between conditions but not by multiple single tests. Therefore, cluster permutations are used.[12]

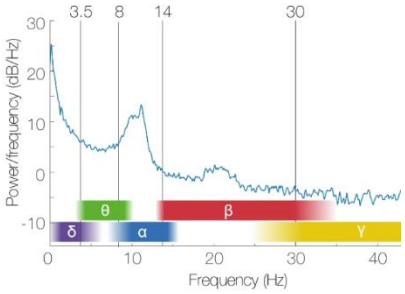
Hypothesis are:
H0: Induced oscillations between conditions does not exist, TFR Difference wave induced = 0
H1: Induced oscillations between conditions exist, TFR Difference wave ≠ 0

[11] https://mne.tools/stable/generated/mne.stats.permutation_cluster_test.html
[12] https://benediktehinger.de/blog/science/statistics-cluster-permutation-test/



T-values and P-Values for the cells and cluster can be observed in the test figure above. **Be careful** here the right two plots are just masks of the left plot.

Results of the cluster permutation test can be seen in the rightmost plot. It found one significant cluster according to the p-value. Therefore the Null-Hypothesis is rejected and **H1** accepted.
Statistically therefore it can be concluded that there is a significant induced effect given alpha = 0.05.
The induced effect is at ~600ms and occurs from low frequencies ~5Hz up to frequencies of ~15Hz.



The conclusion regarding time analysis is that the underlying oscillations of the main experimental contrast were visualized and contain according to the test significant regions from the **theta**, **alpha** and partially also some **beta** waves.

3.6. Decoding analysis

Decode the main contrast of the experiment across time *RQ: When is information about the conditions in our data available?*

In the decoding analysis further analyses of the ERP peak condition is done. The goal is to find points in time where the different conditions rare and frequent are significantly different, which means where their difference holds information.
This task can be formulated as Binary Classification problem, that is the conditions are the two classes to find. As solution to this problem two machine learning algorithms were selected.

1.Logistic Regression:
Logistic regression is good for binary classification (in comparison to linear regression). The logistic sigmoid function is fitted to the data to model the binary threshold between classes.

2.Support Vector Machines (SVMs):
SVMs are a machine learning model that tries to distinguish classes that are initially not linearly separatable and transform them into a high-dimensional feature space using the kernel trick.

Both methods are now applied to the ERP data in the following steps:

Fit Models on epoch data for each subject (09_decoding_01.py)

1. Equalize subject event codes is very important for decoding as there shall be no biases introduced to the model, by the already uneven distribution of events. In the P3 experiment the two conditions are quite uneven in their nr of occurrences. This problem is solved by applying *`epochs.equalize_event_counts(["cond1", "cond2"])`*
2. Binarize the epoch conditions as data labels for our classification step
3. Train/Test splitting

- Apply k-fold cross-validation with k=10 which is a common technique to generalize the accuracy results to different choices of train/test splits → Eliminates the chance to select a specifically good/bad train/test split
- Split data into train and test set with 70% train data and 30% test data for later evaluation

4. Sliding estimator over epoch time allows to fit the model over the epoch time steps
5. Validate the results with cross validation to obtain the scores by calculating the mean over each cross-validation run
6. Subject data and times are saved via Python pickle to be processed later

Aggregate over subjects and calculate statistics (09_decoding_02.py)

1. Load data for all subjects specified via arguments
2. Resample the subjects scores into new sampling rate of scores to smooth the scores
   The sampling precision can be chosen via config
3. Statistically test the classification result score over time:
   Hypothesis here are:
   H0: The classification of the difference in conditions is not greater than random selection, Classification score <= 0.5
   H1: The classification is better than assigning classes randomly, classification score > 0.5

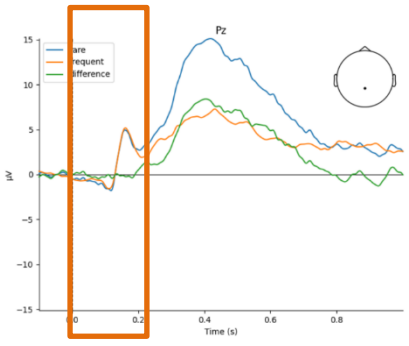
For this a greater-than 1-sample t-test (scipy.stats.ttest_1samp) is applied.
The T-test distinguishes test for statistical significance between:

- sample which is the prediction score from the SVM&LogRegression models in each time step
- population mean random choice between classes = 0.5

Alpha= 5% is chosen as classical significance level

The evaluation of the RQ is shown given the following figures. They show for both ML models the accuracy scores over time aligned over the corresponding t-values over time.
It becomes directly visible that starting at around 200ms the classification result is significant as the t-value falls below the significance level. As the models scores are significantly better than chance up until nearly the end of the effect at ~900ms one can argue that in that time span there lies information between the conditions which can be classified by our models.

Between the two models for the analysis SVM & Logistic Regression there are only few differences. Especially the peak in p-values/drop in score between 0.15s and 0.2s in the Logistic Regression is obvious. When looking back at the ERP it seems reasonable as there is nearly no difference in the average ERP. That means there is no information in that time window that our models could classify, therefore the score there is quite bad and closer to just doing the classification by chance.



Bibliography

[1] M. v. Vliet, Seven quick tips for analysis scripts in neuroimaging, PLoS ComputBiol 16, 2020.

[2] E. S. B. M. A. Widmann, "Digital filter design for electrophysiological data – a practical approach," *Journal of Neuroscience Methods,* vol. 250, pp. 34-46, 2015.

[3] J. L. F. W. Z. A. X. S. S. L. E. S. Kappenman, ERP CORE: An open resource for human event-related potential research, 2020.

[4] R. S. V. J. S. D. Herrmann CS, "Time-frequency analysis of event-related potentials: a brief tutorial.," *Brain Topography,* vol. 27, pp. 438-450, 2014.

[5] I. N. Alain de Cheveigne, Filters: When, Why, and How (Not) to Use Them, Neuron 102, 2019.