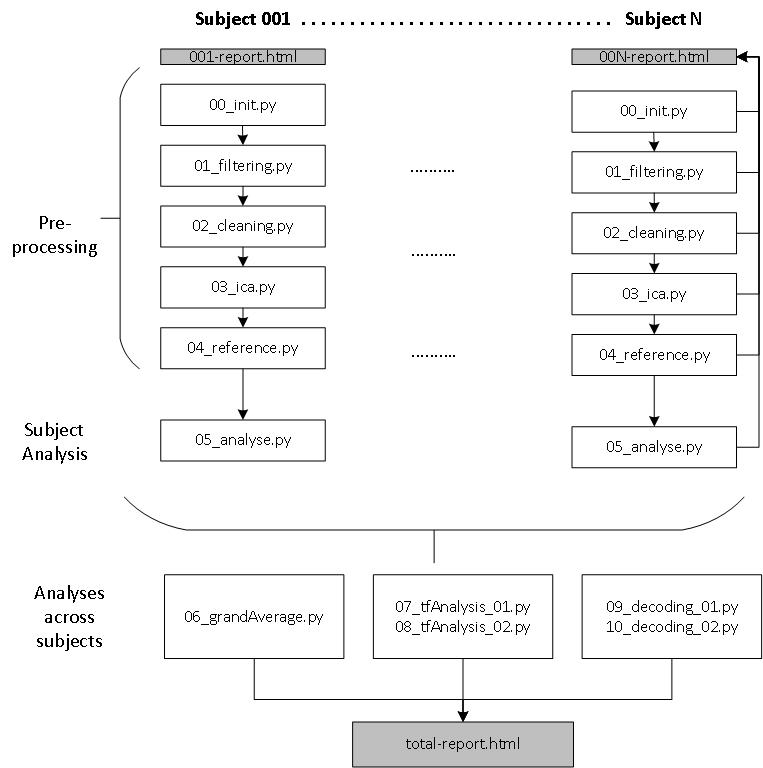
EEG Course: Semesterproject

# Project description

## 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]](#footnote-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]](#footnote-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]](#footnote-3)

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.

* **Reproducability:**

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**

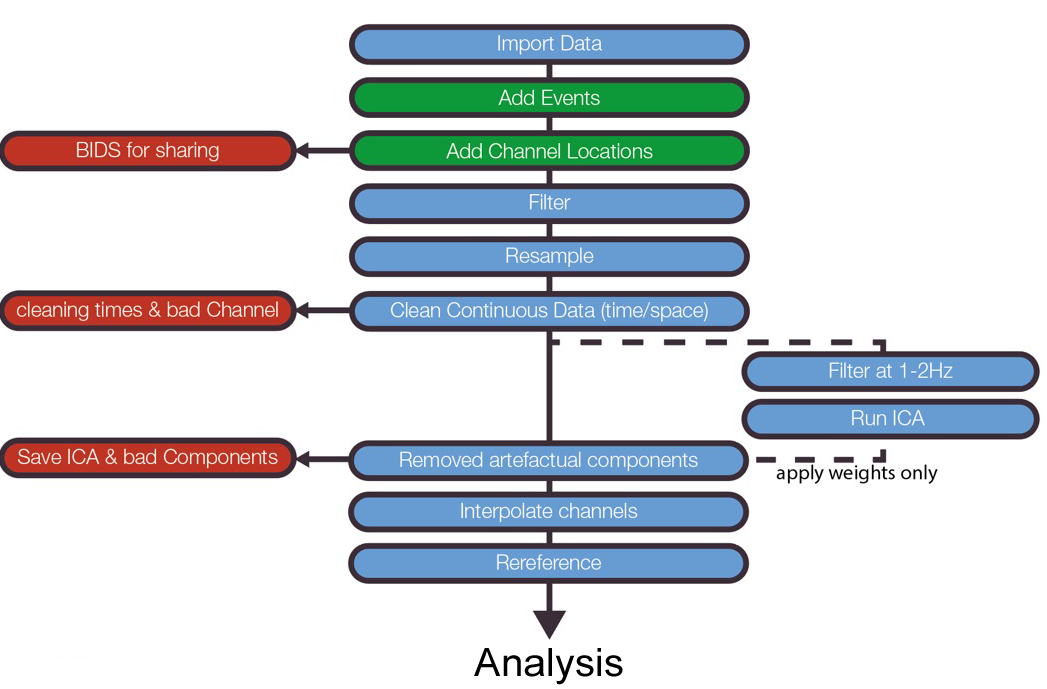
Automatic reports and file generation with the advanced file management system are very handy for analysis purposes. Everything is nicely structured in one place.

## 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 behaviour of the system.

* Config.py:
  + Python file which manages filenames over fnames.py class from conpy[[4]](#footnote-4)
  + Builds YAML config parameters to config object
  + Manages user **path** and threads which has to be set before usage
* Config.yaml:
  + Static config file with relevant parameters over all python files

# Preprocessing



This chart by Cohen shows the basic preprocessing and cleaning steps for EEG data.  
During the preprocessing part I follow this procedure step by step. In the next subchapter the steps are explained in more detail including parameter choices.

## 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 of all filtering is performed. It is important to choose a appropriate filter because filters may cause significant distortions/biases to the data. My chosen therefore are base on a set of best practices an 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. [3] show that high-pass filters with 0.75Hz cut has only minor effects on ERP data. Therefore I choose to stay below this threshold

* **Bandpass-Max:** **50 Hz**

Typical choice for EEG data, Pitfalls of these choices as described in [2]

Following recommendations of Widmann et al. [3]: low-pass filters with cutoff frequencies higher than 40 Hz during ERP analysis are

* **Filter design:** **Firwin**

FIR filters have advantages over IIR filters like [3]:

* + Easier to control
  + Always stable
  + Well-defined passbad
  + Etc.

Additionally [3] recommend to go with a bandpass filter instead of separate low-/highpass filters

## 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”]
  + Interpolates bad channels if there are any

## 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 Highpass Filter as described by Cohen and others.

This shall remove slow drifts in the data before computing the ICA.

Applied Highpass with Cutoff-Frequency 1Hz.

1. Fit the ICA and remove bad components to the initial raw data object
2. Save the new raw object

The implemented ICA has several modes to choose from:

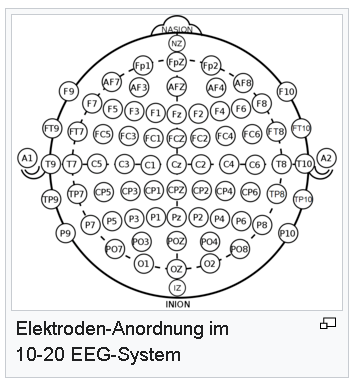
* “PrecomputeMode”: Uses precomputed provided ICA from EEG lab
* “DialogeMode”: Plots the ICA and individual components views directly. This is used for manual cleaning of subjects

## 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-referecenced-raw.fif) |

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 [4] and several Tutorials[[5]](#footnote-5) [[6]](#footnote-6)



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

It was found that there are several possibilities to choose the reference. Modern approaches recommend to go with the Average of the electrodes as Reference, whereas the ERP Core Paper uses the average of the mastoid sites (P9, P10). We also learned that besides these two approaches the REST (Reference Electrode Standardization Techniques) can be used. For this purpose I stayed close to the ERP Core paper by using the local adjacent to the mastoids (average P9/P10) as reference.

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)

1. Generate referencing plots

# Analyse

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

# 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) |

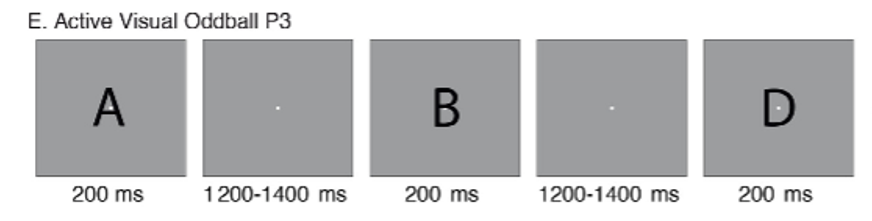
The Grand Average script operates over subjects and computes the evoked difference between the subject conditions. Procedure is as follows

# P300/P3 Analysis: Visual oddball experiment

## 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. [4]

The following figure shows how the experiment looks like:



## 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.

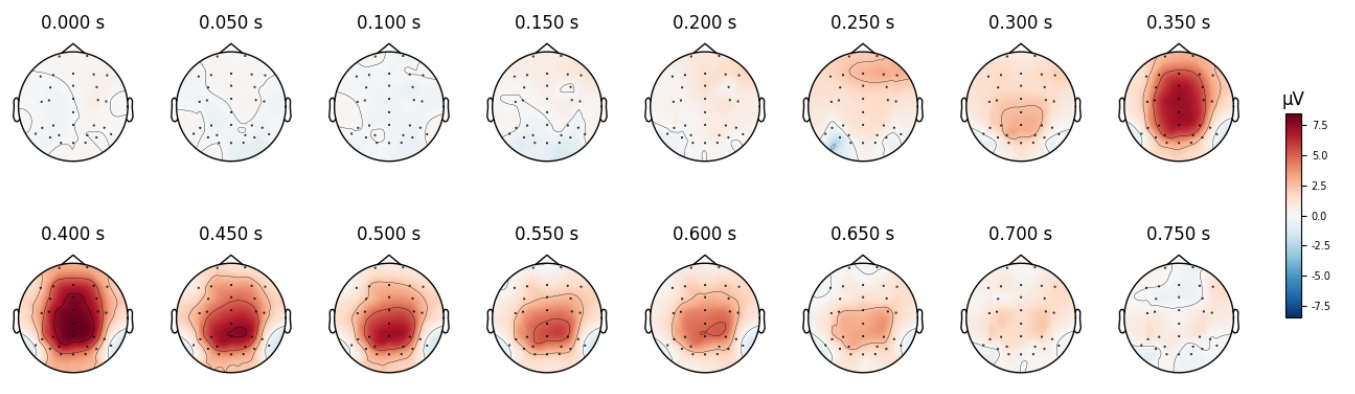
## Manual preprocessing subjects

### Subject 002

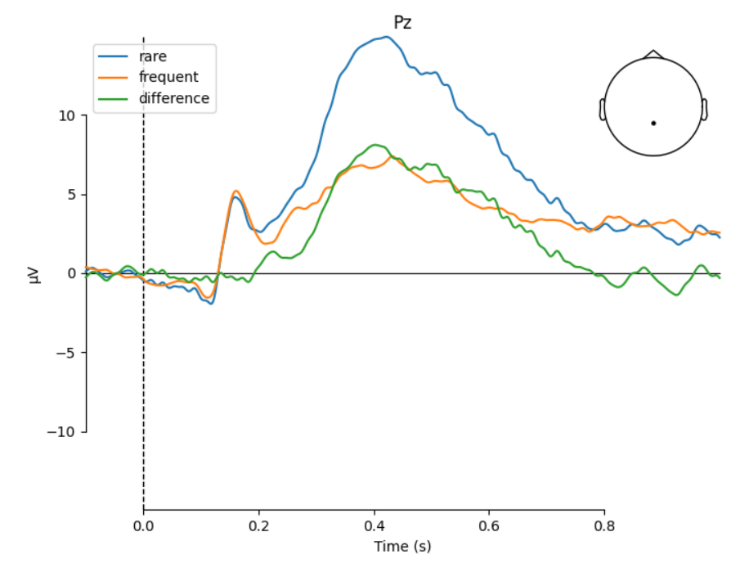
* + 1. Subject 019
    2. Subject 037

## ERP Analysis

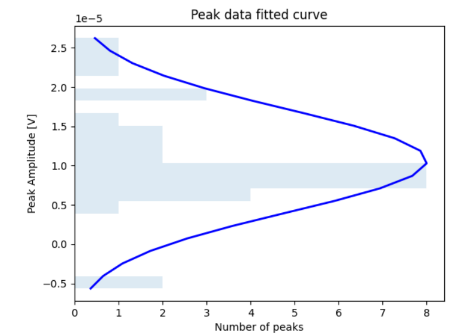
After processing the individual subjects now the analysis of the ERP is done. For this the goal is to find a general significant effect in humans between our two experiment conditions over all the subjects. Therefore one has to compute the grand average over all subjects.  
First for each subject we calculate the difference of the condition evokeds. 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 this conditions. This becomes also visible when looking at the grand average plot in the following figure.

In the topography plot one can already see that there is some kind of artefact starting at ~300ms. The voltage is especially high in the brain midline around electrodes Pz, Cz, CPz.  
For further analysis we now choose the Pz electrode. The reason for this is that we see that it is one of these electrodes with high artefacts. Also in literature the Pz electrode is mostly used for P3 tasks as it is reported in the ERP Core paper [4].

Topography plots for ERP grand average analysis



P3 Grand average of ERP at electrode Pz

Here we can see the voltage curves depending on the condition.  
We now want to test if this artefact is a statistically significant effect.

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. 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 we perform a one-sided T-test using *scipy.stats.ttest\_1samp* which calculates for H0 that the expected value of independent observations is equal to population mean. [[7]](#footnote-7)

our 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:The

According to the T-Test and p-value with alpha =5% we reject the Null-Hypothesis and go with H1. We probably have a statistical significant effect between the conditions given our data evaluated at electrode “Pz” and peaks in time window 300ms-600ms.

# 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] | I. N. Alain de Cheveigne, Filters: When, Why, and How (Not) to Use Them, Neuron 102, 2019. |
| [4] | 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. |

1. <https://pydoit.org/> [↑](#footnote-ref-1)
2. <https://github.com/AaltoImagingLanguage/conpy> [↑](#footnote-ref-2)
3. <https://github.com/wmvanvliet> [↑](#footnote-ref-3)
4. <https://github.com/AaltoImagingLanguage/conpy> [↑](#footnote-ref-4)
5. <https://mne.tools/stable/auto_tutorials/preprocessing/plot_55_setting_eeg_reference.html> [↑](#footnote-ref-5)
6. [https://www.fieldtriptoolbox.org/faq/why\_should\_i\_use\_an\_average\_reference\_for\_eeg\_source\_reconstruction](https://www.fieldtriptoolbox.org/faq/why_should_i_use_an_average_reference_for_eeg_source_reconstruction/) [↑](#footnote-ref-6)
7. <https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_1samp.html> [↑](#footnote-ref-7)