MNE-Python
-
introduction to EEG data analysis

25th-26th November 2024
CIMCYC, University of Granada

Ivan Padezhki & Rodika Sokoliuk

Welcome

Slot 1: Monday, 15-16:30

Course program

Open Science

Introduction into Python

Course Program

⏰	MNE-Python Workshop		Deep Neural Networks with Python
	Monday, 25th November	**Tuesday, 26th November**	**Wednesday, 27th November**
9:30 - 11:30		Resolution of day 1 issues & Preprocessing 1 (theory & hands-on) SLIDES 18 - 28	Principles of Deep Neural Networks & Introduction to *PyTorch* package
11:30 - 12:00		*COFFEE BREAK*	*COFFEE BREAK*
12:00 - 14:00	Optional: Installing MNE-Python and Visual Studio Code	Preprocessing 2 (theory & hands-on) ERP analysis (theory & hands-on) SLIDES 29 - 38	Example of Neural Network Design, Implementation, and Training
14:00 - 15:00	*LUNCH BREAK*	*LUNCH BREAK*	
15:00 - 16:30	Welcome & Getting to know (MNE-)Python SLIDES 3 - 12	Time-frequency analysis (theory & hands-on) SLIDES 39 - 46	
16:30 - 17:00	*COFFEE BREAK*	*COFFEE BREAK*	
17:00 - 18:30	How to prepare an EEG experiment & How do EEG data look like (theory & hands-on) SLIDES 13 - 17	Working on your own code OR How to convert data formats into BIDS SLIDES 47 - end	

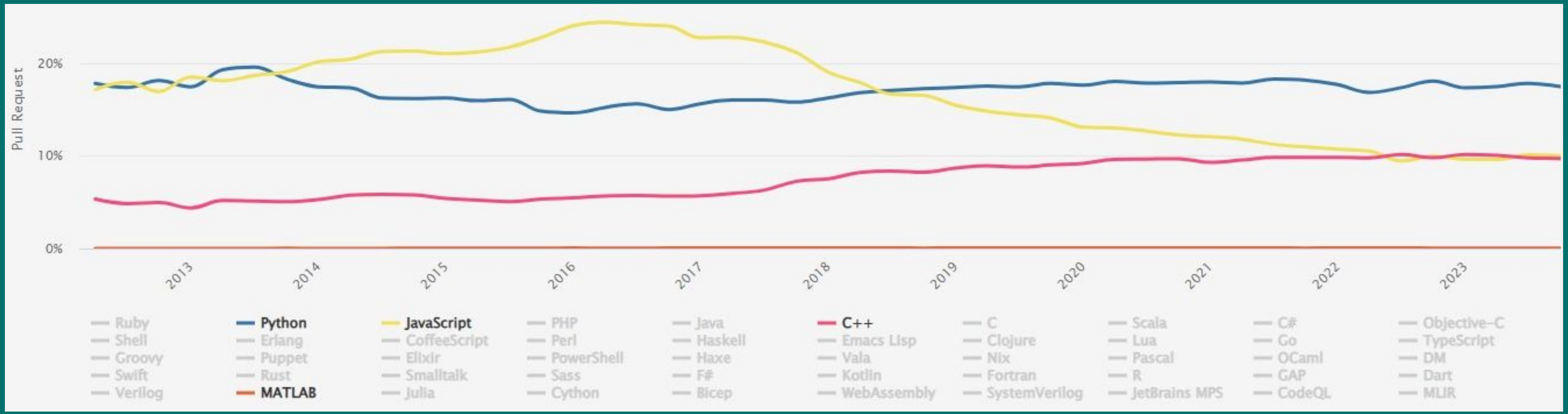
Course Program

What will I learn?

⇒ basics about MNE (how to load and visualise data, how to preprocess data and how to conduct more advanced analyses)

Why should I learn this?

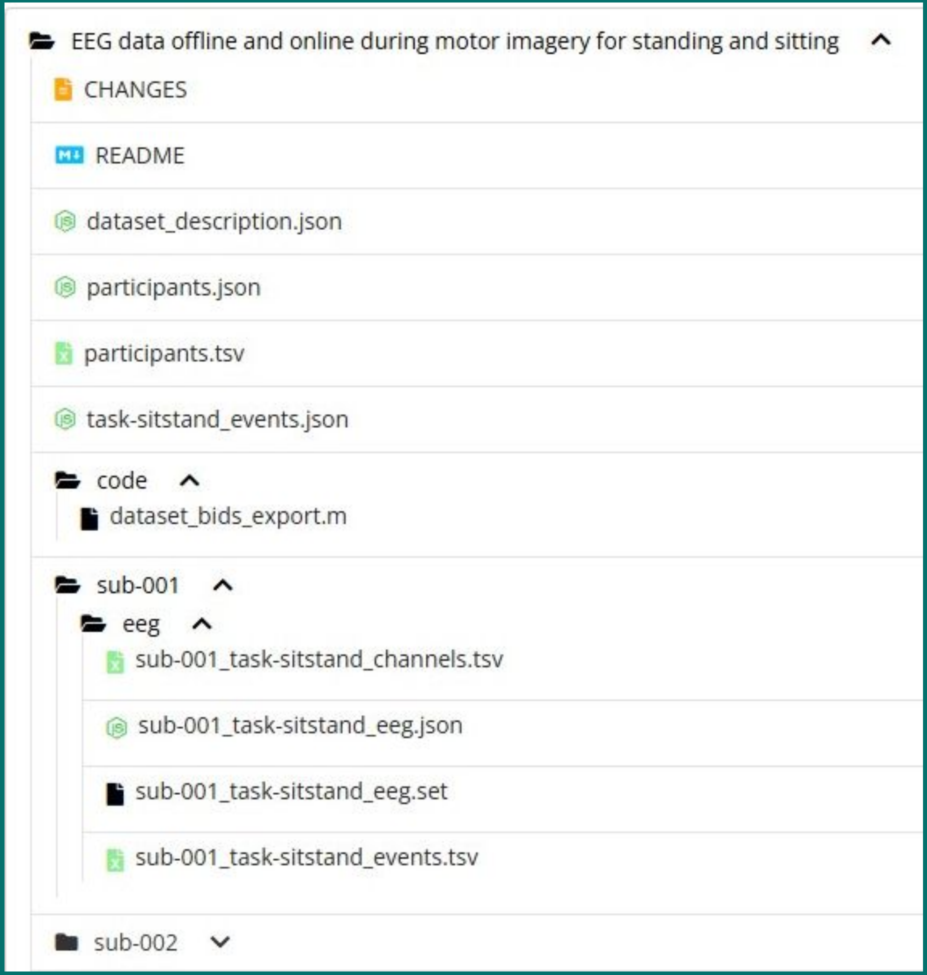
⇒ Python & MNE Python are **open-source** softwares with a vibrant community



Open Science

Python as a step to open science

- FAIR principle: Findable, Accessible, Interoperable, Reusable
- BIDS format
- version control systems (GitHub, GitHub Copilot)



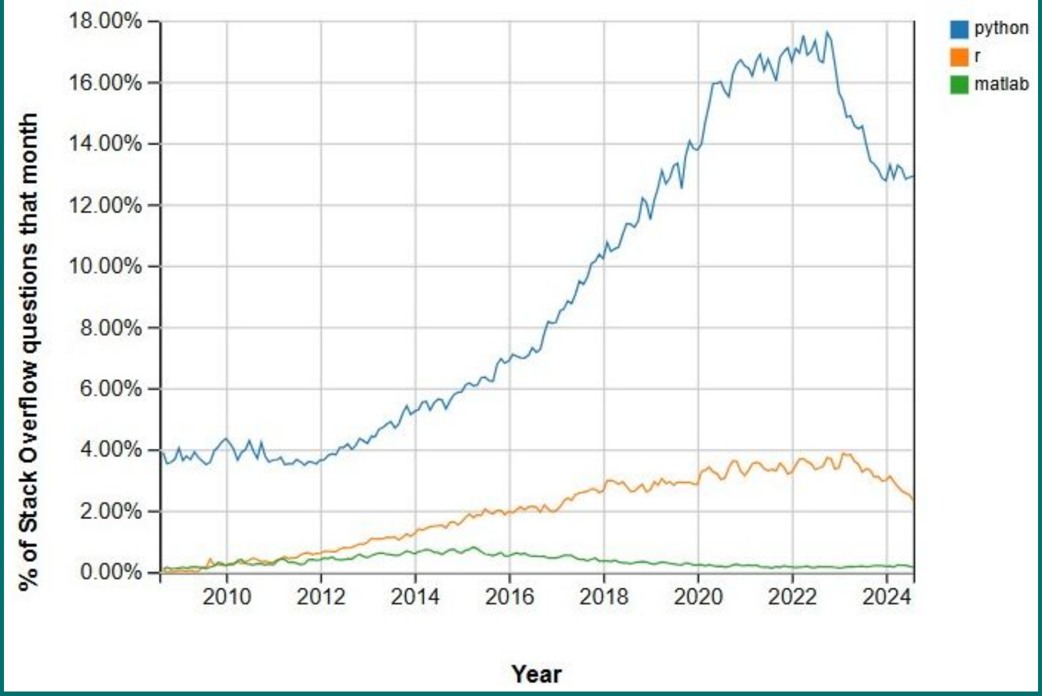
Introduction to Python

How do I learn?

By treating programming in Python as the target of your problem-solving skills and curiosity as a researcher

Where do I find support?

- ChatGPT
- GitHub Copilot
- we will share useful links with you!



Introduction to Python

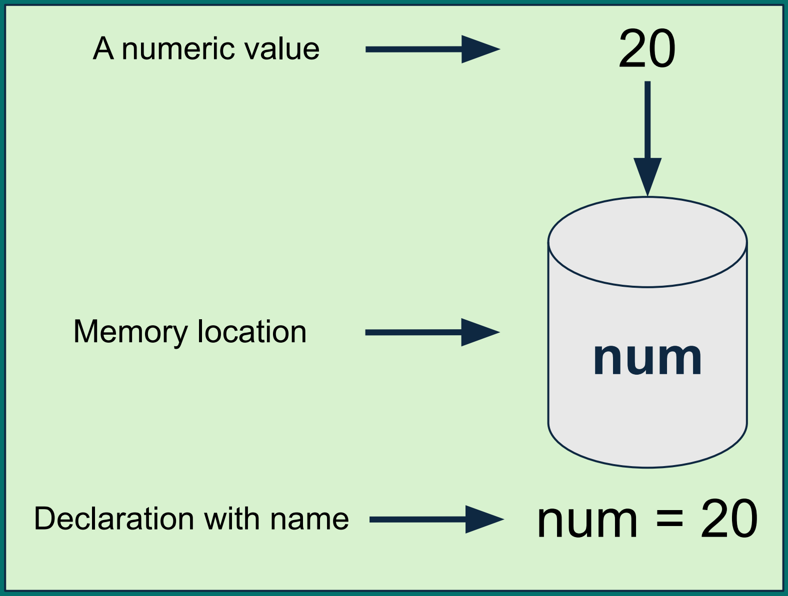
Everything is an object:

6, “my_name”, the function print(), a subject’s dataset

- each object type has specific properties
- Style Guidelines for Naming, Spacing, etc.

A variable is a container with:

- name
- location
- value
- type



Basic variable types

- Int, float, string, bool

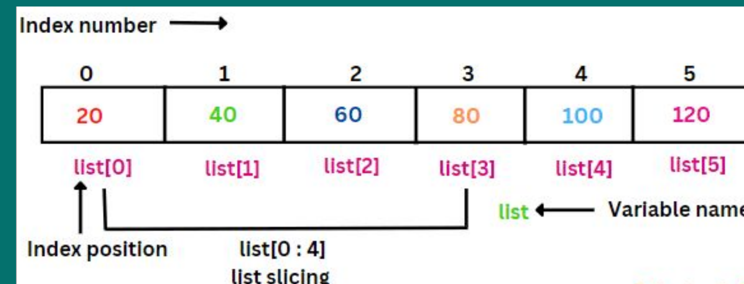
Introduction to Python

Collecting objects together:

- **List** = [ordered, changeable, not unique]
- **Tuple** = (ordered and unchangeable, not unique)
- **Set** = {unordered, not indexed, unchangeable, unique}
- **Dictionary** = {
  ‘ordered’: False,
  ‘mutable’: True,
  ‘iterable’: ‘yes’
  }

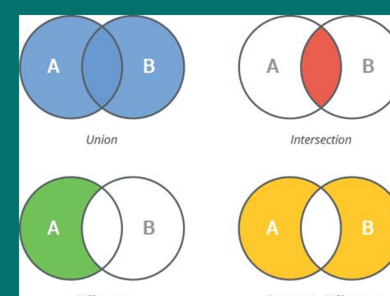
dict.keys()

dict.values()



Python Tuple Operations

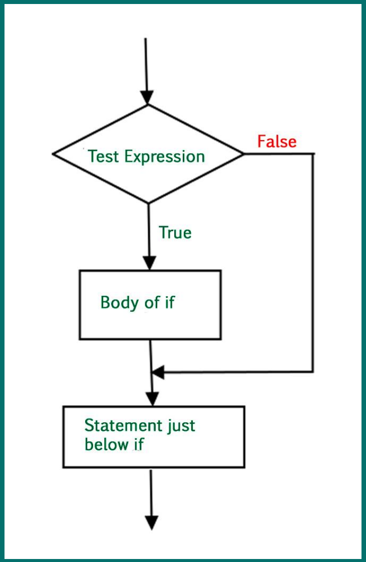
Python Expression	Results	Description
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6,)	Concatenation
(‘Oh’,) * 3	(‘Oh’. ‘Oh’, ‘Oh”)	Repetition
“the” in (‘a’, ‘an’, ‘the’)	True	Membership
for x in (1, 2, 3, 4): print (x),	1234	Iteration



Introduction to Python

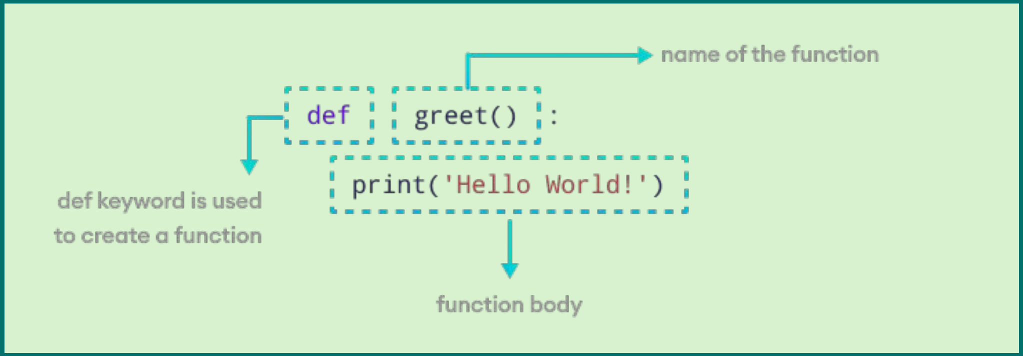
Control Flow:

- **If / Elif / Else** – indentation matters
- **For** – iterate over lists, arrays from 0 to n-1 elements
- **While** – when we don't know the end condition



Functions:

- **Parameters** – free or default
- **Docstrings** – document what your function does



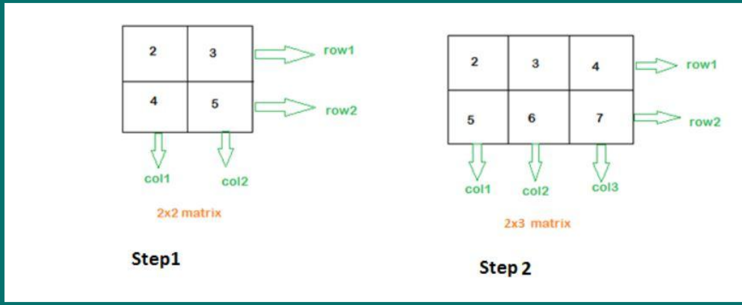
Python: no longer basics

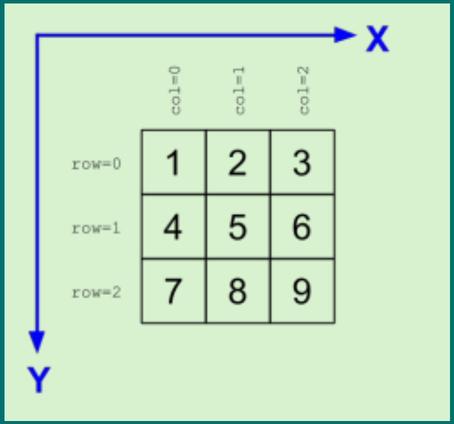
NumPy: a library for fast operations on multidimensional data

Arrays – hold our data
- We can access individual elements – or select rows, slices from arrays
- Broadcasting: element-wise operations can get funny when arrays have different lengths

Matrices – are just arrays
- Reshape
- Transpose
- Multiply – the dot product

Boolean indexing (Masking) – what we use to find data and select it based on conditions

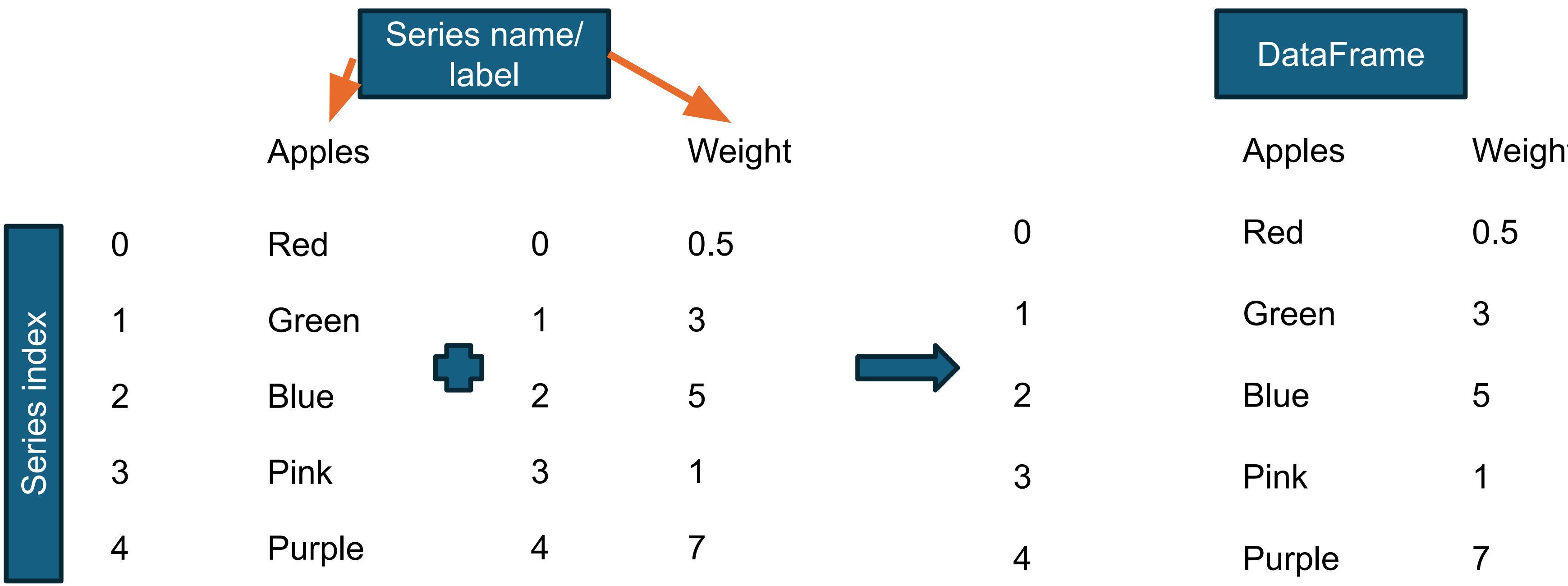




Python: no longer basics

Pandas: a library for manipulating structured data

- **Series:** One-dimensional labeled array: a list with labels
- **DataFrame:** Two-dimensional table with labeled columns (axis 1) and indexed rows (axis 0)



Slot 2: Monday, 17-18:30

Theory part: Good EEG practice & how your data should look like (and also how it shouldn't)

Hands-on part: Looking at your data in MNE

Optional: "Homework"

Good EEG practice

How do you design your experiments?

Design and record so you can
- match your preregistration/ analysis plan
- Reuse the data

- ❑ Optimise recording settings
- ❑ Complete & self-sufficient metadata
- ❑ Montage
- ❑ Sampling rate
- ❑ Reference electrodes
- ❑ Triggers, timings, events
- ❑ Beginnings, interruptions, endings of blocks and recording

```
# Define new keys for event IDs
mapping = {
    # current
    "encoding/current/1": 111,
    "encoding/current/2": 121,
    "encoding/current/3": 131,
    "encoding/current/4": 141,

    "cue/current/1": 113,
    "cue/current/2": 123,
    "cue/current/3": 133,
    "cue/current/4": 143,

    "probe1/current/1": 115,
    "probe1/current/2": 125,
    "probe1/current/3": 135,
    "probe1/current/4": 145,

    "probe2/current/1": 117,
    "probe2/current/2": 127,
    "probe2/current/3": 137,
    "probe2/current/4": 147,
```

```
    "enc_cue_fixation/delayed/1": 12,
    "enc_cue_fixation/delayed/2": 22,
    "enc_cue_fixation/delayed/3": 32,
    "enc_cue_fixation/delayed/4": 42,

    "first_jitter/delayed/1": 14,
    "first_jitter/delayed/2": 24,
    "first_jitter/delayed/3": 34,
    "first_jitter/delayed/4": 44,

    "second_jitter/delayed/1": 16,
    "second_jitter/delayed/2": 26,
    "second_jitter/delayed/3": 36,
    "second_jitter/delayed/4": 46,

    "iti_jitter/delayed/1": 18,
    "iti_jitter/delayed/2": 28,
    "iti_jitter/delayed/3": 38,
    "iti_jitter/delayed/4": 48,

    # other triggers
    "stop_recording": 255,
    "new_segment": 99999
}
```

Good EEG practice

Think before you start collecting data

⇒ Specific analyses require specific data (e.g., if you're interested in oscillations, your analysis windows have to show a specific length)

Document every step of data collection

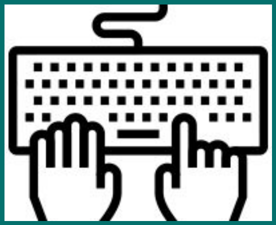
sub-01	**F1** noisy in block 1	participant had to go to the **bathroom after block 3**
sub-02	**FC3 and FC2** are pulling throughout full experiment	participant complains the **cap was not comfortable**
sub-03	no noisy channels	
sub-04	first 4 blocks had **many noisy channels**	participant arrived with **damp** hair
sub-05		

Hands-on: Looking at data in MNE

Input: Supported data formats

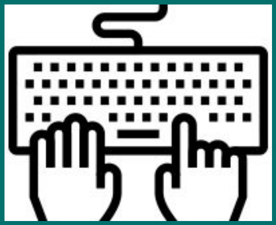
- **Raw data**:
  Brainvision, Biosemi, Neuroscan CNT, European data format (EDF), EGI simple binary, EGI MFF format, eXimia, General data format, Nicolet, Persyst
- **Preprocessed data**:
  Fieldtrip, EEGLAB

☞ **Or simply: transform to EEG-BIDS**



Homework

Try what we did today (loading and visualising data, inspection of artefacts etc.)



Slot 3: Tuesday, 09:30-11:30

Questions & comments on day 1 content

Theory part: Preprocessing part 1
(artefacts, bad channels, filtering)

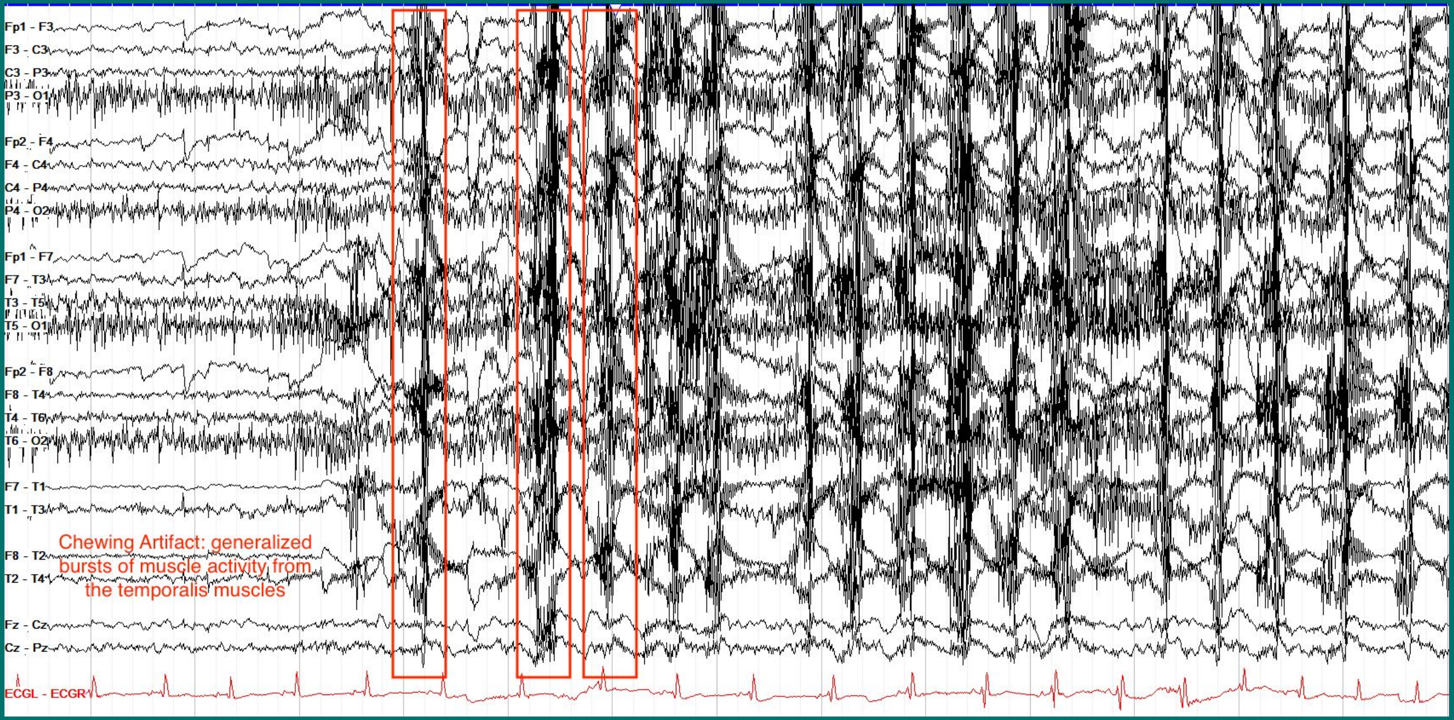
Hands-on part: Preprocessing part 1
(loading data, visual inspection & artefact rejection, filtering, visualisation)

Questions & Comments on day 1

Theory: Preprocessing 1

Artefacts

Manual versus Automated?



Theory: Preprocessing 1

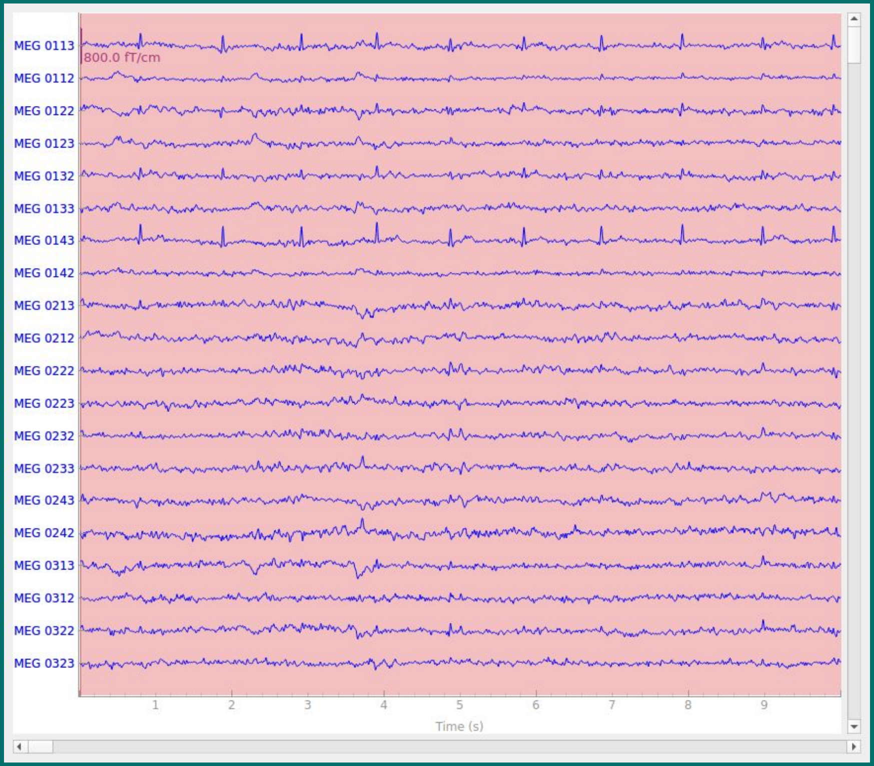
Automated break detection

MNE-output:

Detected 2 break periods of >= 25 s duration:

0.0 – 51.4 s [51.4 s]
185.6 – 277.7 s [92.1 s]

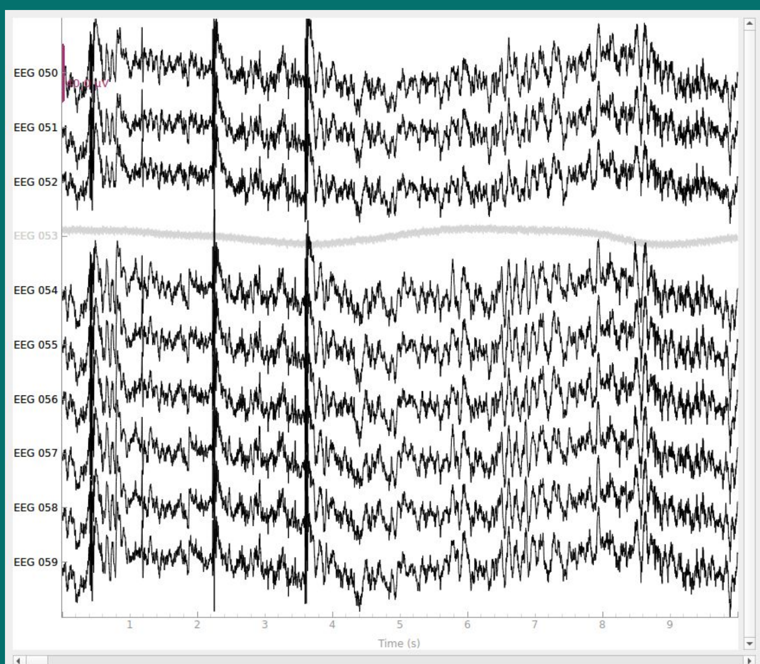
In total, 51.7% of the data (143.5 s) have been marked as a break.

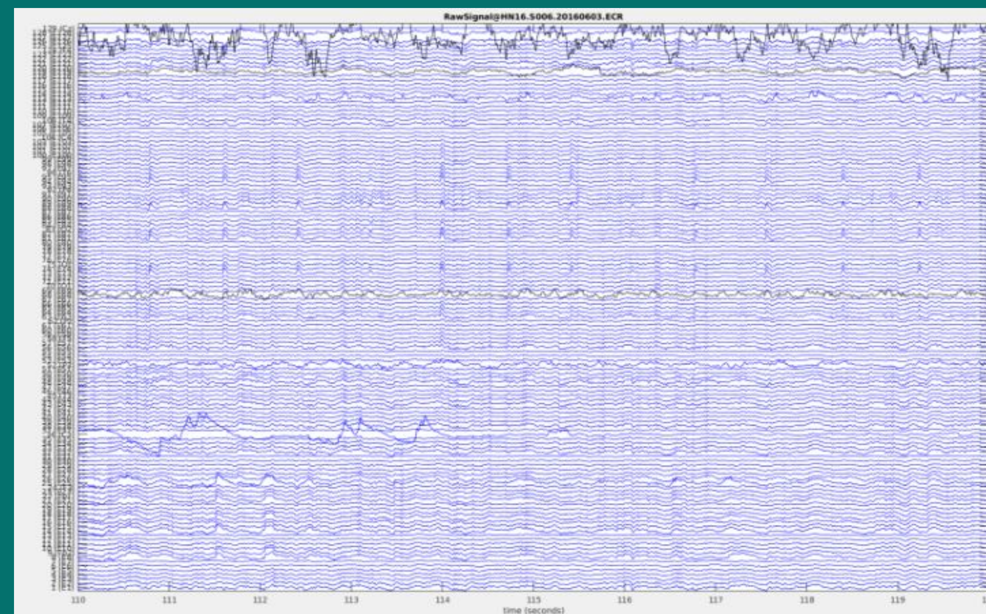


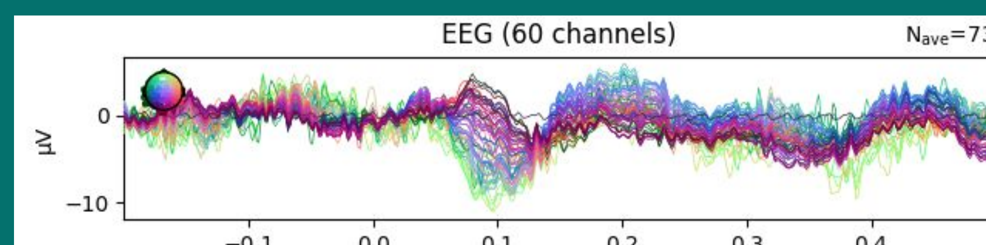
Theory: Preprocessing 1

Bad channels

come in many shapes and sizes







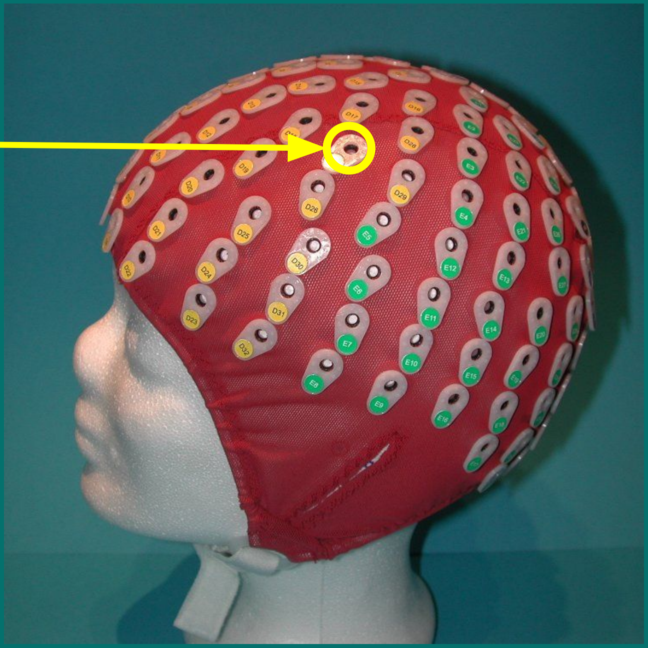
Theory: Preprocessing 1

Bad channels: Interpolation

Reconstructing lost data by interpolating data over “clean” neighbouring channels

Important for group analyses ⇒ same number of channels

flat, noisy,
or
“jumping”



Theory: Preprocessing 1

Bad channels: Interpolation

Reconstructing lost data by interpolating data over
“clean” neighbouring channels

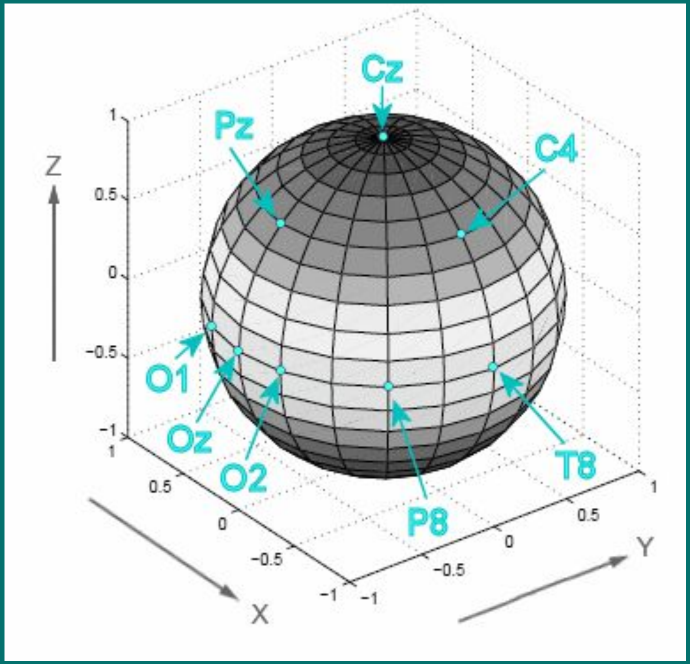
Important for group analyses ⇒ same number of channels

reconstructing
lost data

Theory: Preprocessing 1

Bad channels: Interpolation

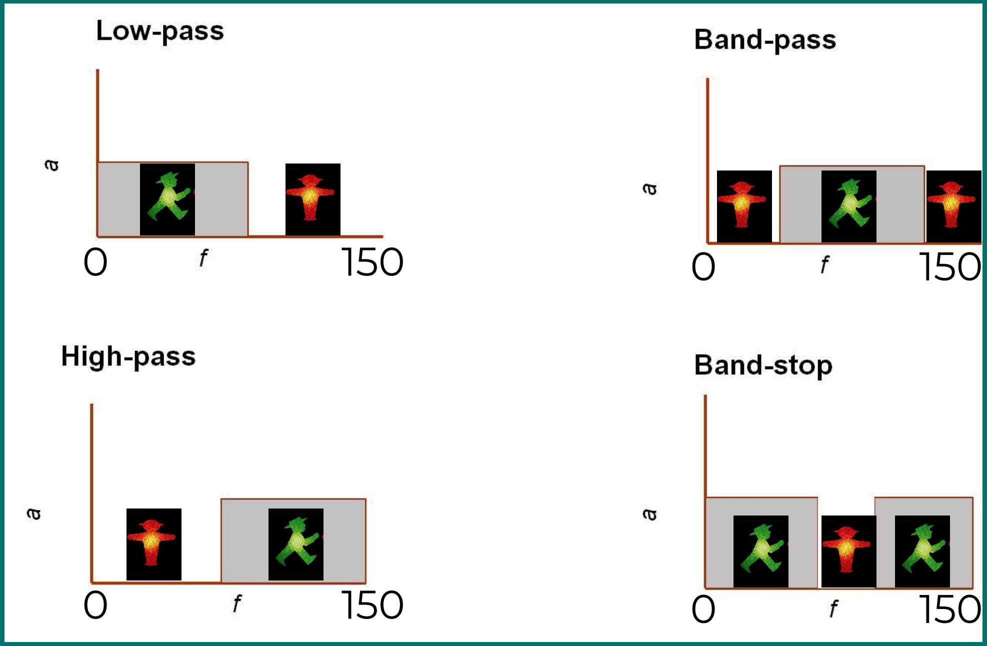
Spherical spline method: take surrounding good signal and combine to replace bad one



Theory: Preprocessing 1

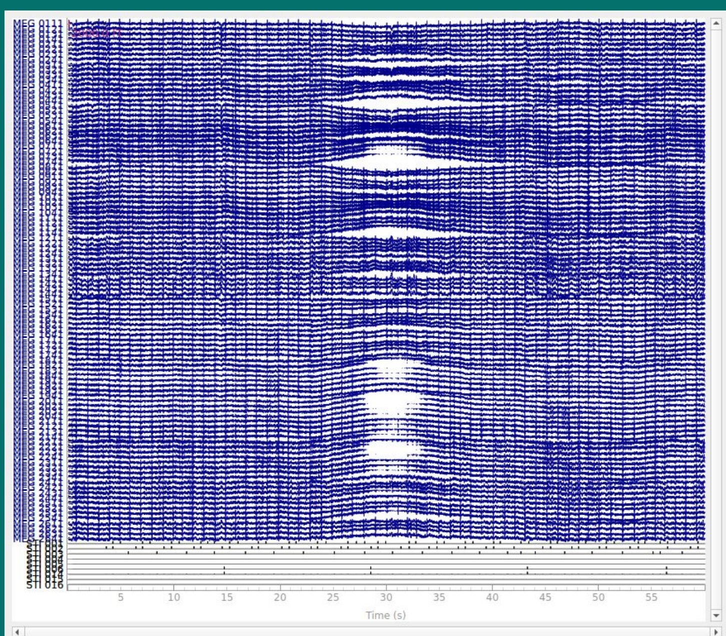
Filtering

- **Attenuates or removes** parts of the signal
- **Report:** type, cut-off frequency, length
- **Common use:** slow drifts and power line noise



Theory: Preprocessing 1

Filtering: e.g. High-pass against slow drifts

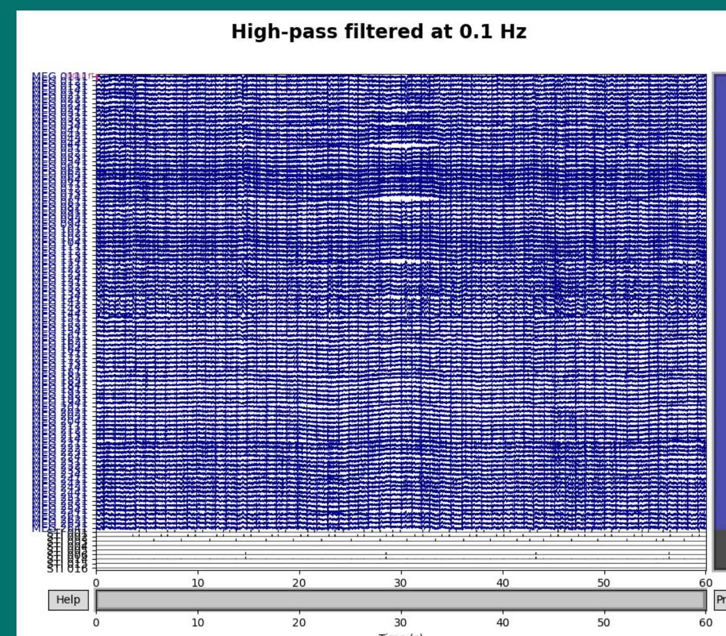


MNE-output:

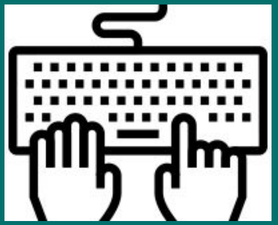
Filtering raw data in 1 contiguous segment
Setting up high-pass filter at 0.1 Hz

FIR filter parameters

Designing a one-pass, zero-phase, non-causal highpass filter:
- Windowed time-domain design (firwin) method
- Hamming window with 0.0194 passband ripple and 53 dB stopband attenuation
- Lower passband edge: 0.10
- Lower transition bandwidth: 0.10 Hz (-6 dB cutoff frequency: 0.05 Hz)
- Filter length: 19821 samples (33.001 s)



Hands-on: Preprocessing 1



Slot 4: Tuesday, 12:00-14:00

Theory part: Preprocessing part 2
(triggers & epoching, ICA, re-referencing)

Hands-on part: Preprocessing part 2
(triggers & epoching, ICA, re-referencing, plotting)

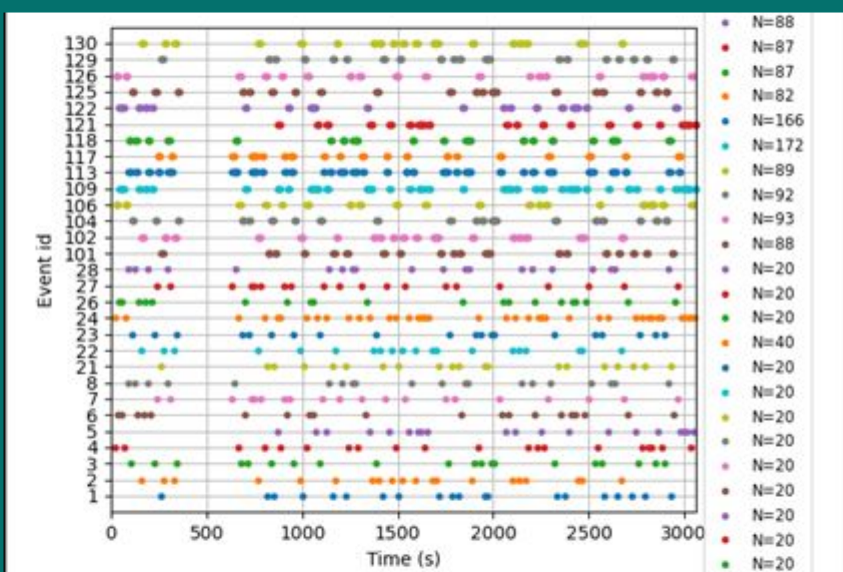
Theory part: ERP analysis
(What is this and what do we use it for?)

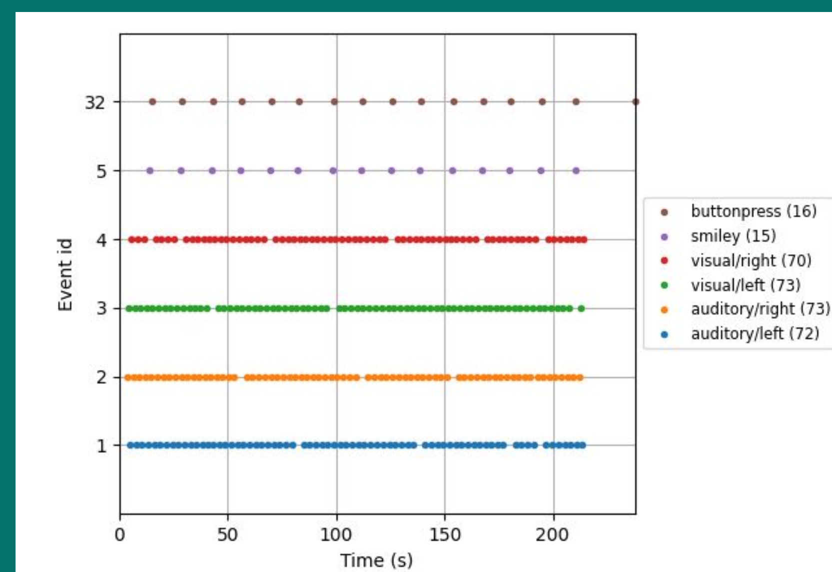
Hands-on part: ERP analysis
(ERP analysis in MNE, computing differences between conditions, cluster test for testing statistical significance)

Theory: Preprocessing 2

Triggers to Events

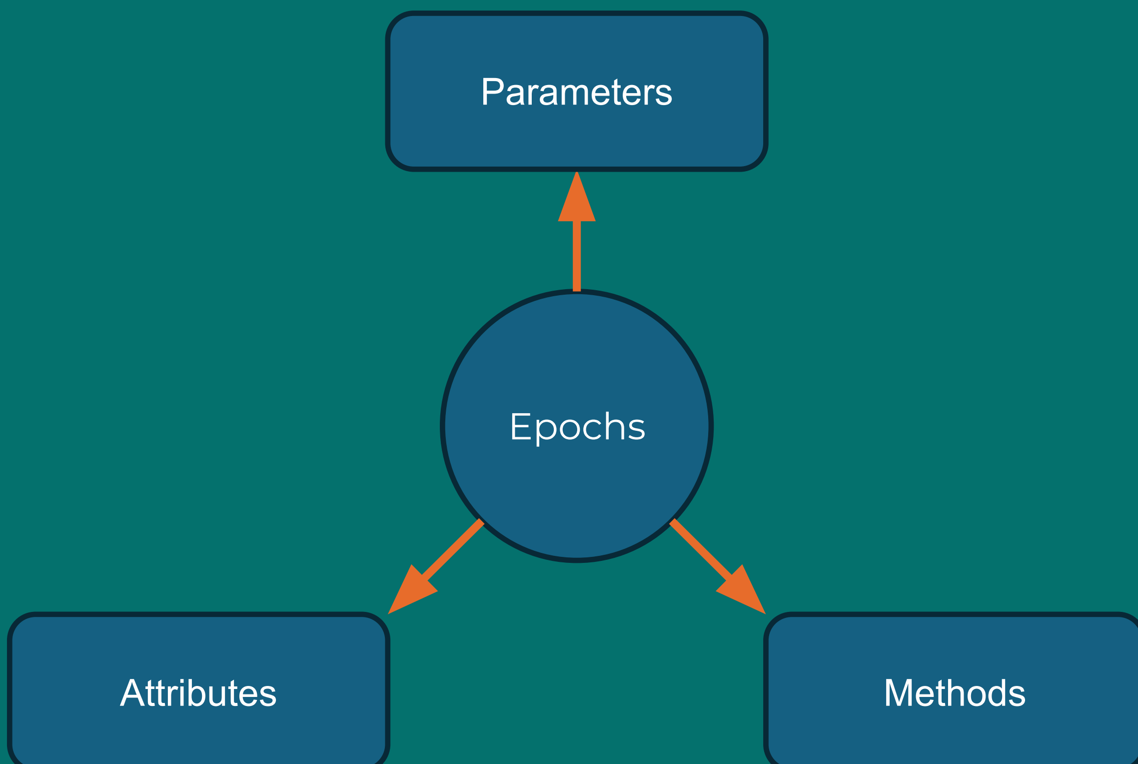
Event dictionary **maps** list of numerical trigger IDs to informative event names





Theory: Preprocessing 2

- Raw signal
- Events, IDs
- Timings
- Metadata
- Baseline



- .info
- .events
- .event_id
- .drop_log
- .times
- .metadata

- .apply_baseline()
- .apply_hilbert()
- .average()
- .copy()
- .drop()
- .plot()

Theory: Preprocessing 2

Epochs and their metadata

- A Pandas DataFrame
- Conditions, responses, reaction times

Labels

	WORD	Concreteness	WordFrequency	OrthographicDistance	NumberOfLetters	BigramFrequency
0	film	5.450000	3.189490	1.75	4.0	343.250
1	cent	5.900000	3.700704	1.35	4.0	546.750
2	shot	4.600000	2.858537	1.20	4.0	484.750
3	cold	3.700000	3.454540	1.15	4.0	1095.250
4	main	3.000000	3.539076	1.35	4.0	686.000

Stimulus

Epoch number

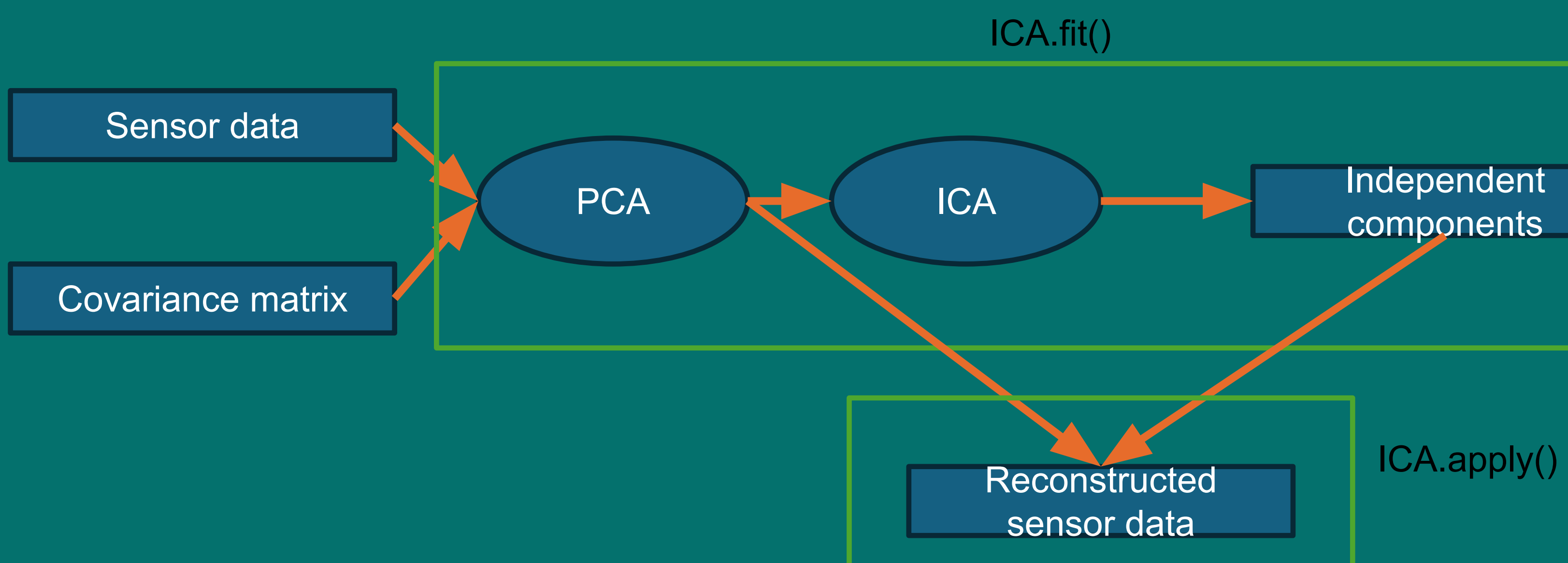
Stimulus characteristics

Theory: Preprocessing 2

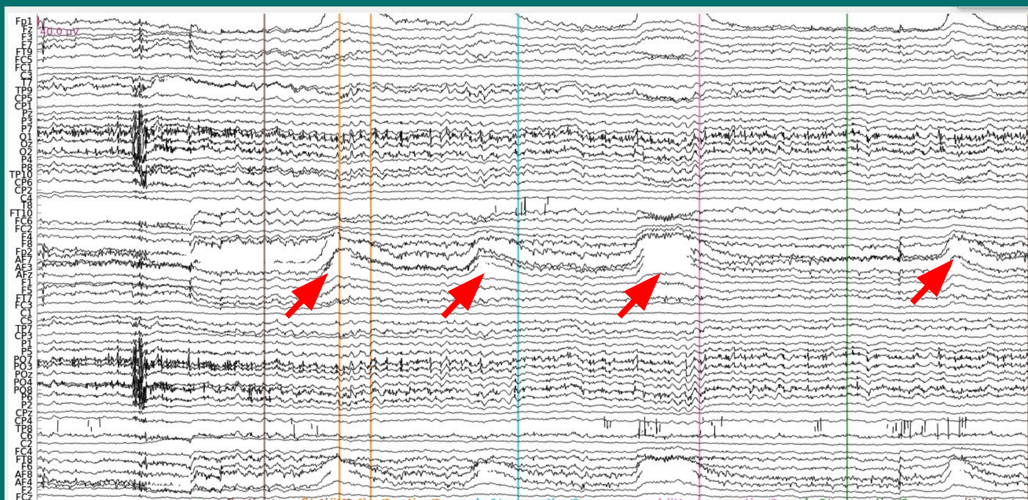
ICA (Independent Component Analysis)

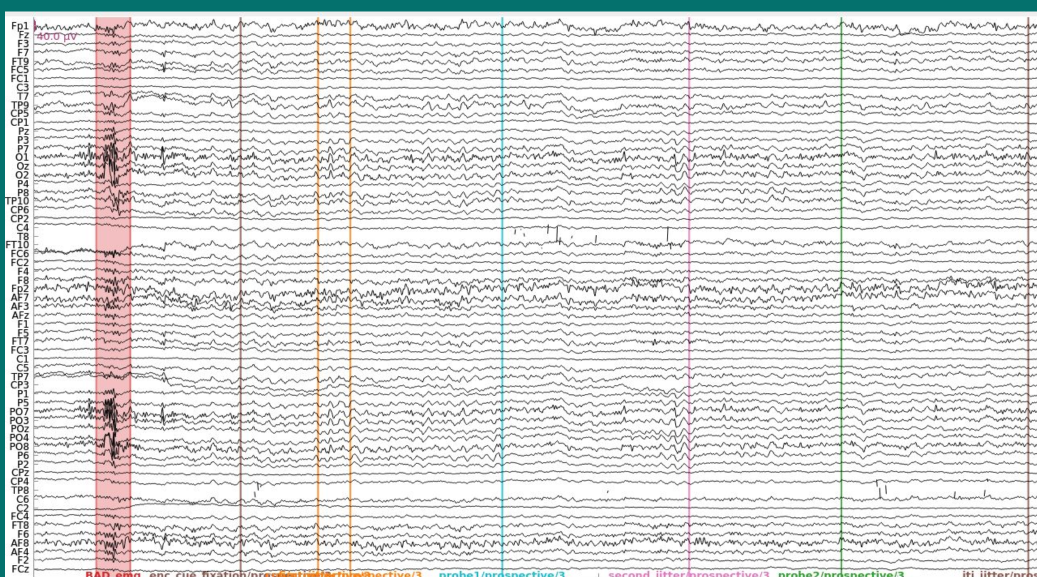
- Independent source components in the signal
- Great for **estimating** and **excluding** blinks and eye movements

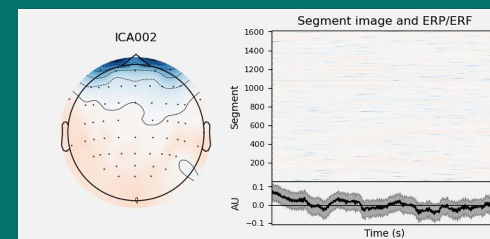
ICA is a Python object:

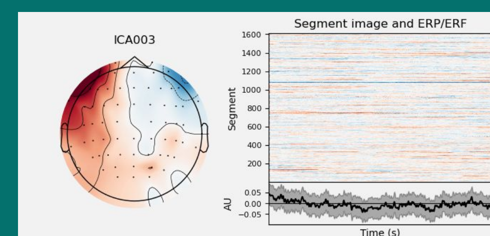


Theory: Preprocessing 2





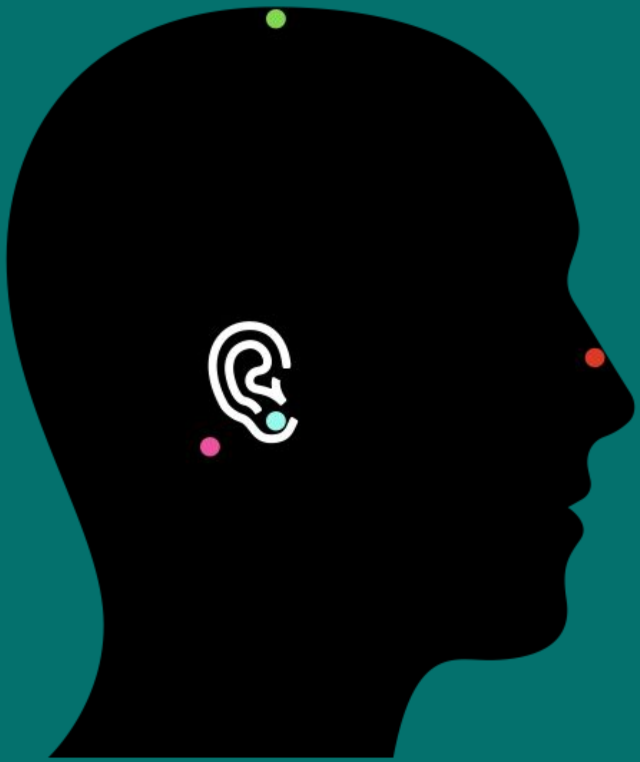




Theory: Preprocessing 2

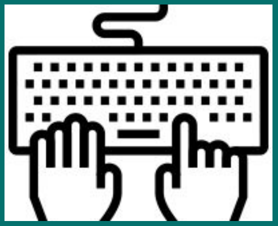
Re-referencing

In every EEG recording there has to be a reference electrode (typically on sites that are negligible, like mastoids, nose tip, earlobes, vertex). Re-referencing the data post-hoc can reduce noise and can help with signal interpretation (e.g., re-referencing to average => improves spatial specificity)



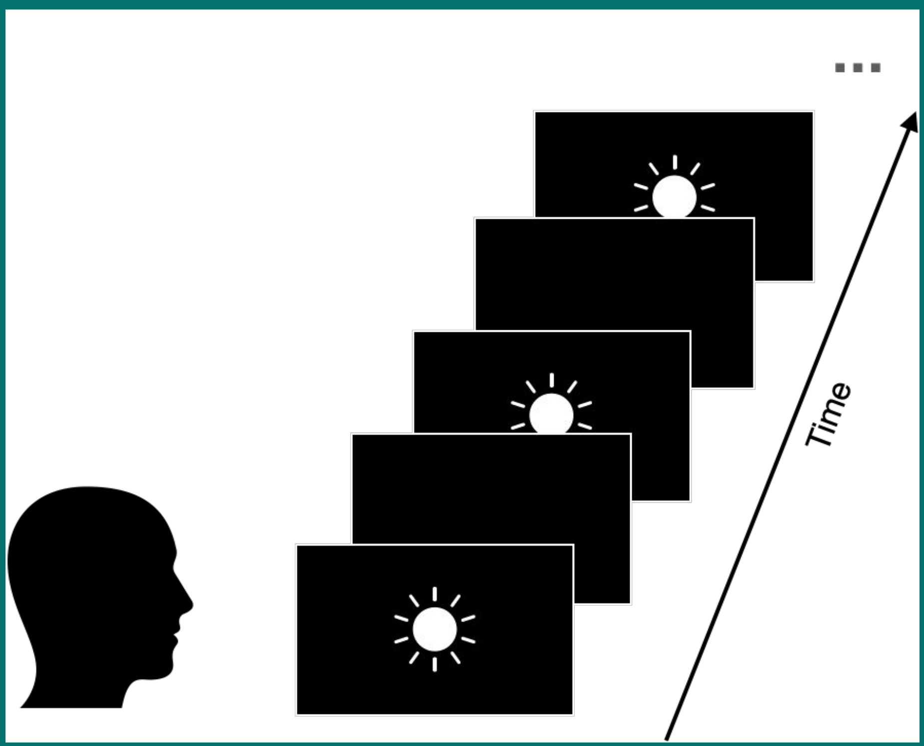
nose
mastoid
earlobe
vertex

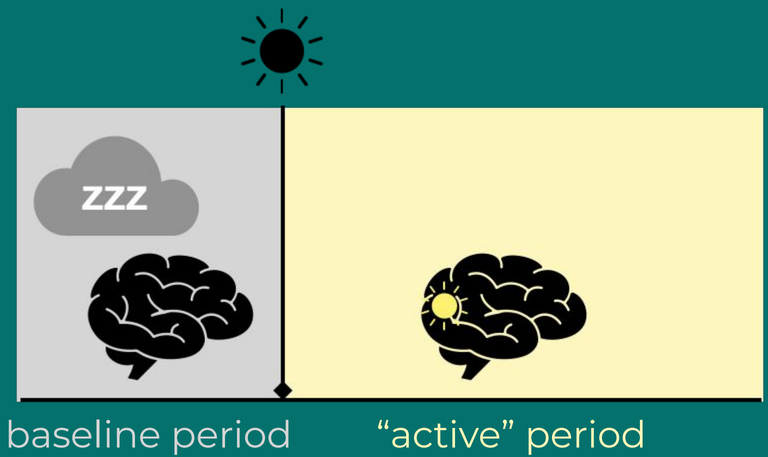
Hands-on: Preprocessing 2



Theory: ERP analysis

ERP = Evoked Response Potential

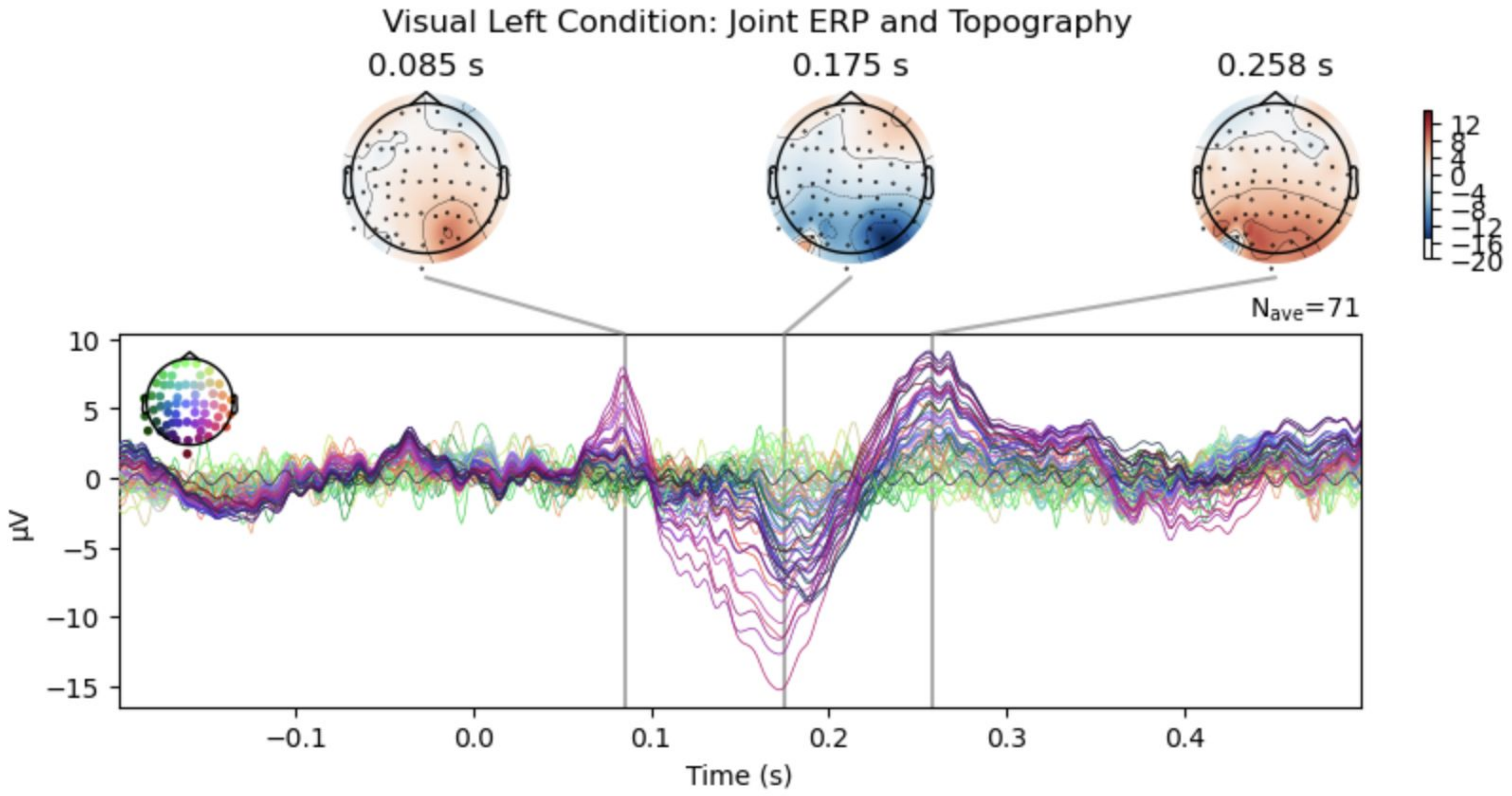




ERP = "active" vs. baseline period

Theory: ERP analysis

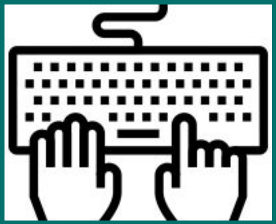
ERP = Evoked Response Potential



Hands-on: ERP analysis

Computing ERPs in MNE

Computing ERP differences between 2 conditions



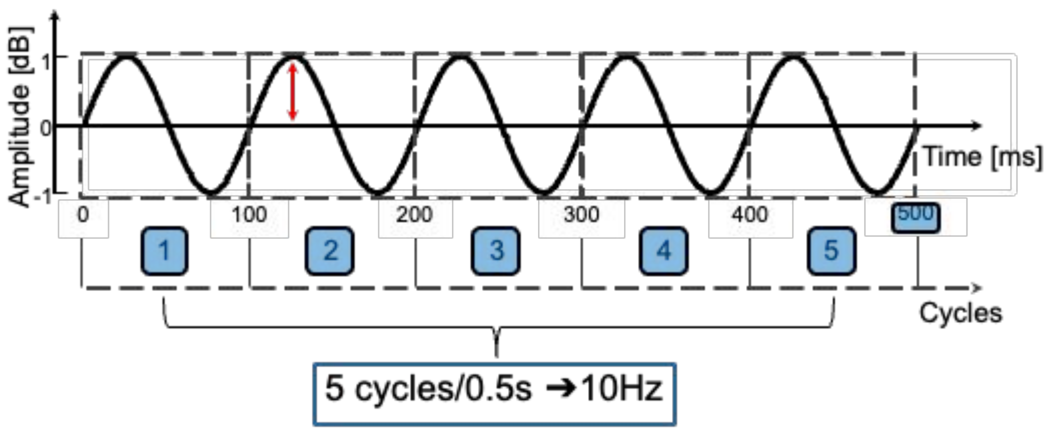
Slot 5: Tuesday, 15:00-16:30

Theory part: Time-frequency analysis
(characteristics of brain oscillations, different ways to analyse oscillatory brain activity)

Hands-on part: Time-frequency analysis
(Different options of time-frequency analysis using MNE, cluster test for testing statistical significance)

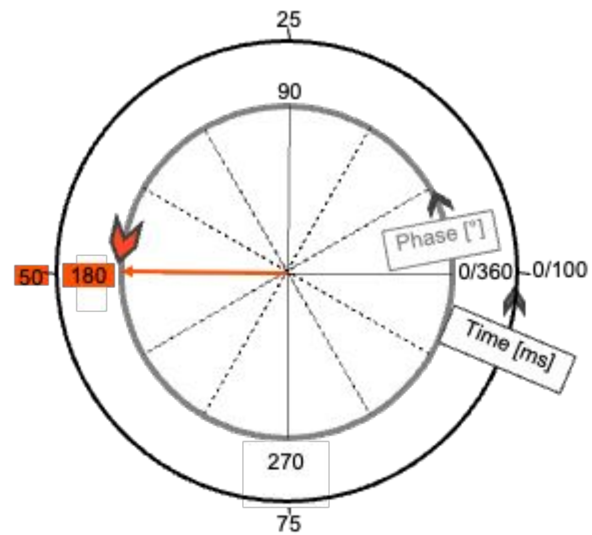
Theory: Time-frequency analysis

Brain oscillations



Phase: 0 - 2π / 0 – 360°
➔ changes over time

Frequency = number of cycles/1s



Theory: Time-frequency analysis

Brain oscillations

Brain oscillations were discovered with the invention of the electroencephalography (EEG)

First human EEG recording, by Hans Berger, 1928 (alpha rhythm)

Theory: Time-frequency analysis

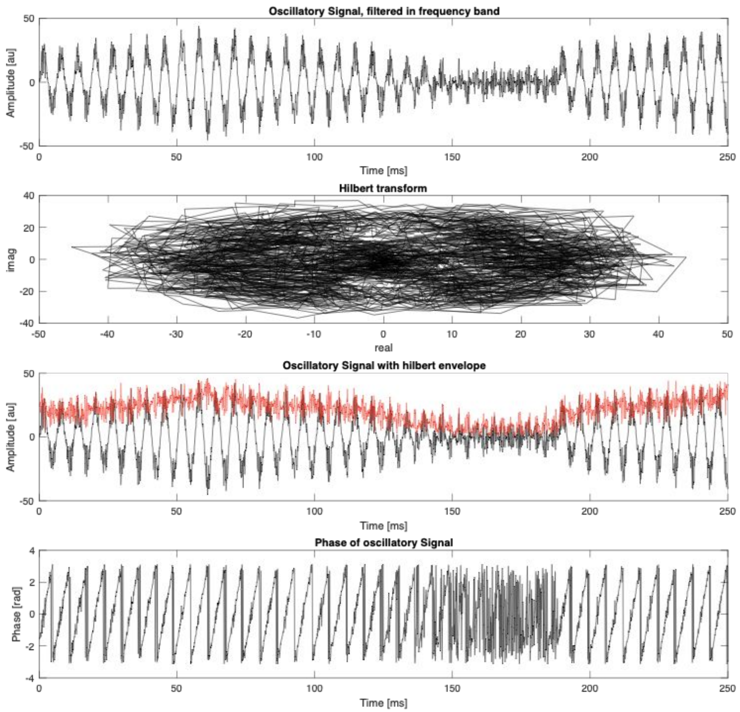
Different approaches:
Filtering + Hilbert (for a specific frequency band only)

filtered signal

hilbert transform

signal with hilbert envelope

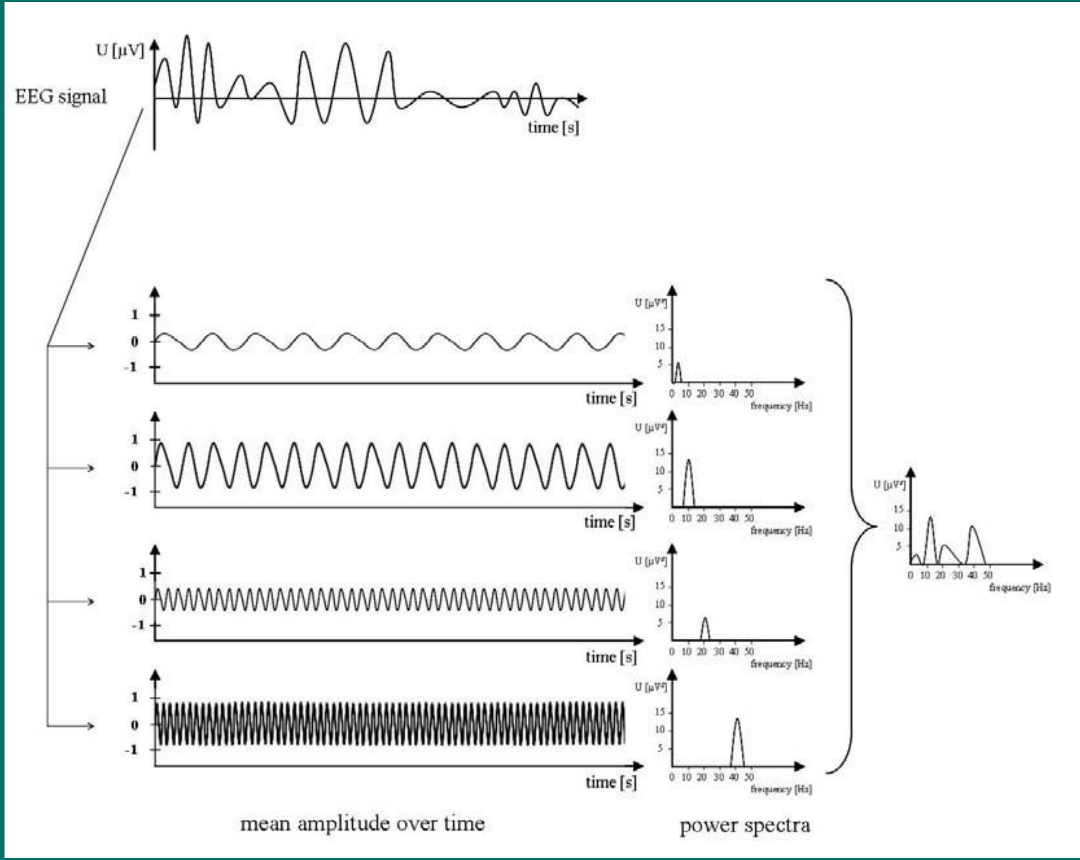
phase of signal



Theory: Time-frequency analysis

Different approaches:
Fast Fourier analysis

Decomposition of overall EEG signal per electrode in different frequency bands.
Mean amplitude of computed frequency = contribution of frequency to global EEG waveform.
Power spectra = proportional distribution of all frequency bands to EEG signal.
No temporal information!

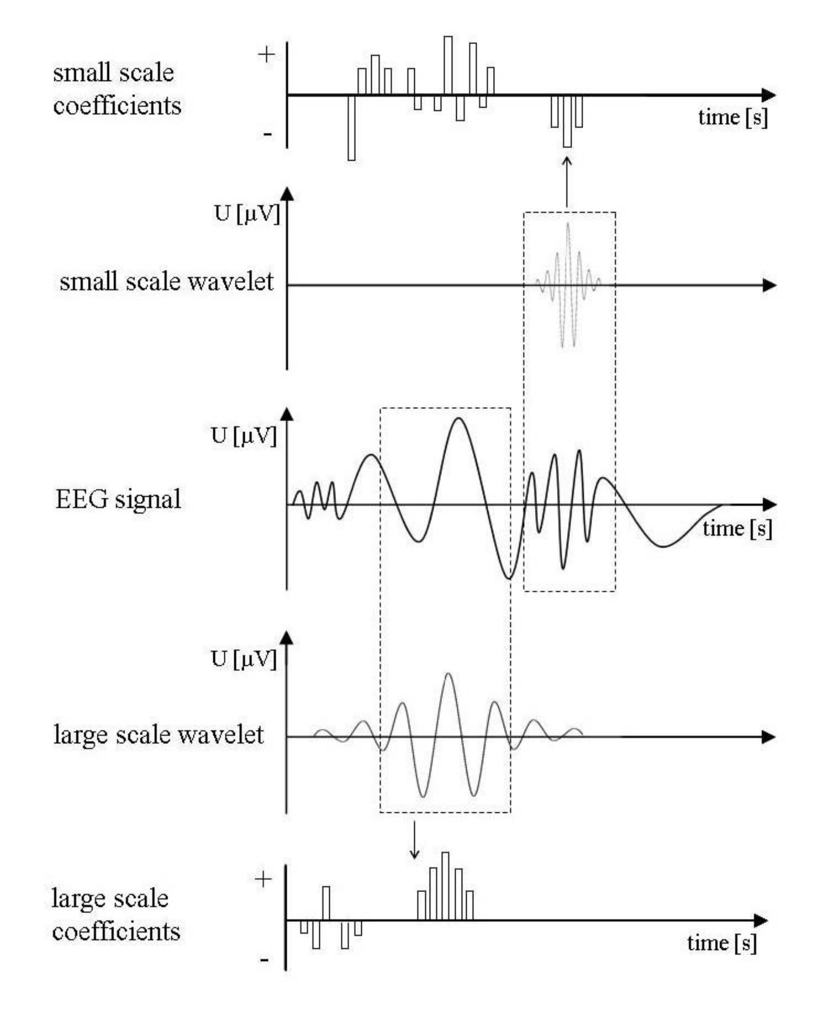


adapted from Jobert, 2002

Theory: Time-frequency analysis

Different approaches: Morlet wavelets

Depending on whether high or low frequency arrays of the EEG signal are to be detected, small or large scale wavelets move along the time course of the EEG waveform. Positive coefficients are computed for matches between the wavelet and the corresponding part of the waveform. Negative coefficients represent mismatches.

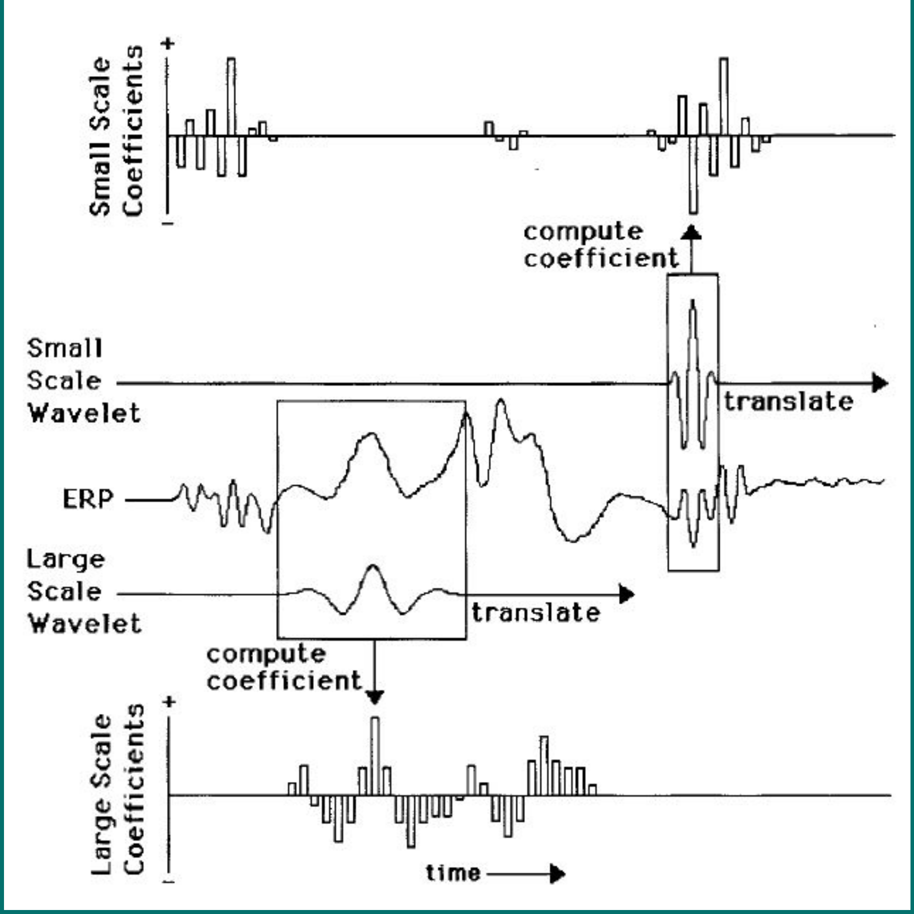


adapted from Samar et al., 1999

Theory: Time-frequency analysis

Different approaches: Morlet wavelets

Depending on whether high or low frequency arrays of the EEG signal are to be detected, small or large scale wavelets move along the time course of the EEG waveform. Positive coefficients are computed for matches between the wavelet and the corresponding part of the waveform. Negative coefficients represent mismatches.

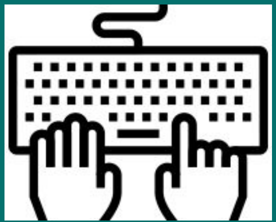


Samar et al., 1999

Hands-on: Time-frequency analysis

Different approaches in MNE:
Wavelets, Fast Fourier, Filtering + Hilbert, etc.

⇒ Wavelets



Slot 6: Tuesday, 17:00-18:30

Hands-on part: Working on your own on more advanced analyses using your data or MNE datasets

OR

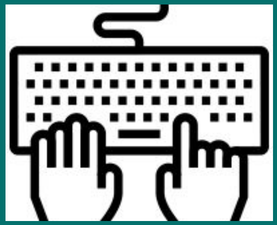
Hands-on part: How to convert data formats into BIDS in MNE

Hands-on: Working on your own 1

Analyse your own data

Try what we did using your own data (or any sample dataset of MNE):

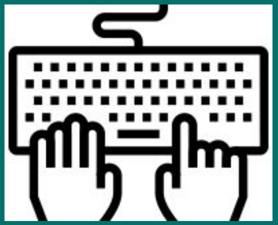
From pre-processing up to time-frequency and ERP analyses. Feel free to use the code we've used and adapt where necessary.



Hands-on: Working on your own 2

Source estimation

Try to visualise the results of the time-frequency and ERP analyses on the source level. Adapt the following steps to fit with the dataset.



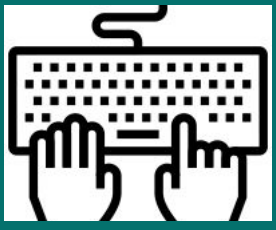
Hands-on: Convert data into BIDS format

BIDS conversion

```python
new_path = "/mnt/NeuroNas/ivan/InstrAct_EEG/data/raw"
# change subject number here:
bids_root = op.join(new_path)

# Datatype might also be inferred automatically, but it's safer to specify it
bids_path = BIDSPath(
    subject=sub_num,
    datatype="eeg",
    task="instract",
    root=bids_root
)
# Add extensions to the path
bids_path.update(suffix="eeg", extension=".vhdr")
# Power line frequency missing in raw.info
raw.info["line_freq"] = 50  # required by BIDS. 50Hz in Spain

# Create bids structure for first participant
write_raw_bids(
    raw,
    bids_path=bids_path,
    events=events,
    event_id=mapping,
    overwrite=True,
    verbose=True,
)
```

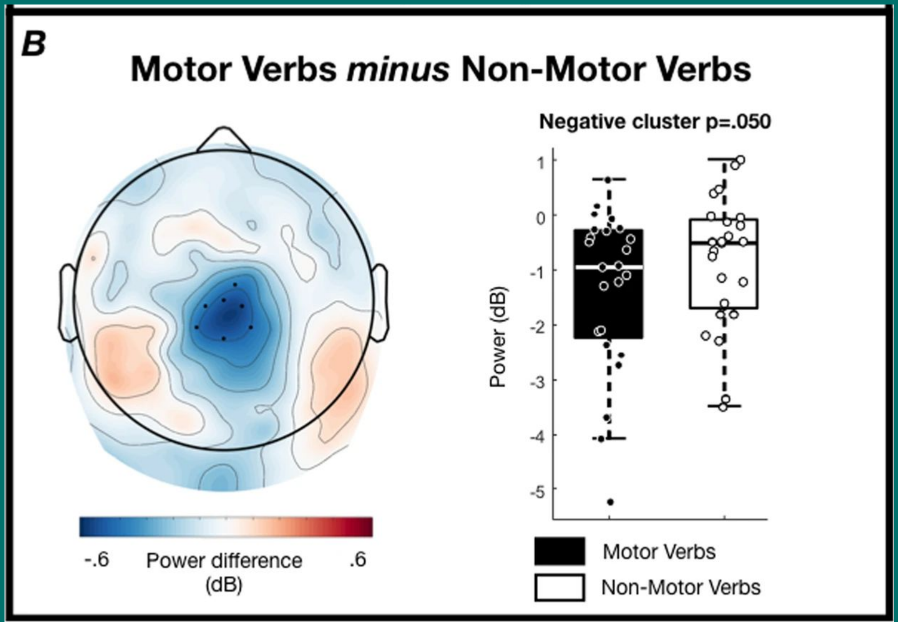


Theory: Cluster-based permutation test

EEG signal: time points x electrodes

⇒ many datapoints ⇒ risk of false positives

Cluster-based permutation tests to test for adjacent datapoints (neighbouring electrodes or timepoints) that show significant difference between conditions



Theory: Cluster-based permutation test

What does it do? (example: paired-sample t-test)

- Computing test at each electrode and time point (or averaged over time period)
- Checking for neighbouring electrodes that show significant difference (using a p-threshold; e.g., p=0.05) which build a “cluster”
- Building the sum over the test statistics of all points within a cluster
- Permutation: Randomly assign labels of the 2 conditions to the data, run the same statistical test (t-test, then look for clusters) and repeat this a lot of times(~1000x)
- Compare cluster stats against permutation distribution (if cluster stats > 5% (depends on p) of permutation distribution ⇒ significant cluster)

Theory: Cluster-based permutation test

Code example

```
# Run the analysis
T_obs, clusters, cluster_p_values, H0 = permutation_cluster_1samp_test(
    epochs_power,
    n_permutations=n_permutations,
    threshold=t_thresh,
    tail=tail,
    adjacency=adjacency,
    out_type="mask",
    verbose=True,
)
```