

Analyzing the time-frequency content of EEG data

Alexander Enge

Neuro Lab @ Humboldt-Universität zu Berlin

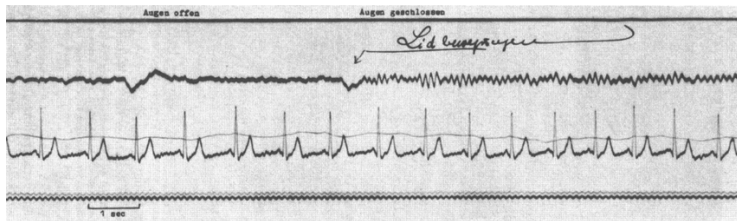
25/05/2022

Today

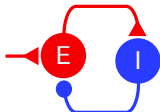
- ① Why study neural oscillations?
 - Evoked vs. induced activity
- ② Frequency analysis
 - Sine waves, Fourier transform
- ③ Time-frequency analysis
 - Morlet wavelets
- ④ Example workflow
 - MNE-Python style, hu-neuro-pipeline style

Why study neural oscillations?

Empirical data



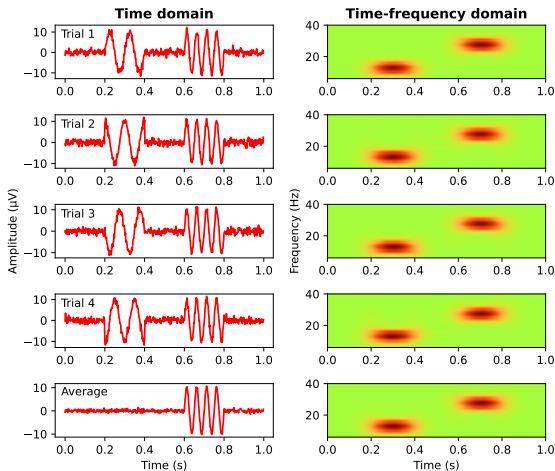
Algorithm



Computation

- Working memory
- Language
- Consciousness

Evoked vs. induced activity

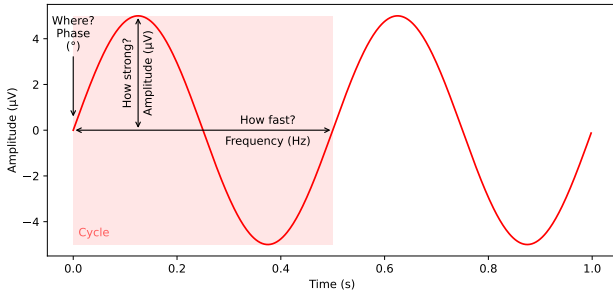


Frequency analysis

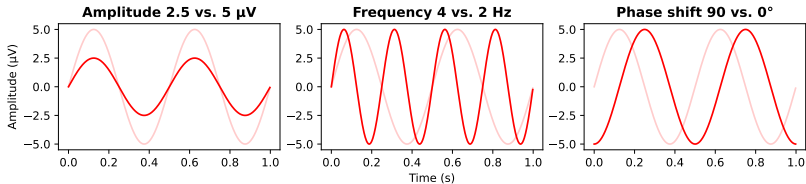
Goal: Examine which frequencies (oscillations) contribute to a stretch of continuous EEG

Approach: Decompose the continuous EEG into a set of sine waves → Fourier transform

Sine waves



Sine waves



Fourier transform



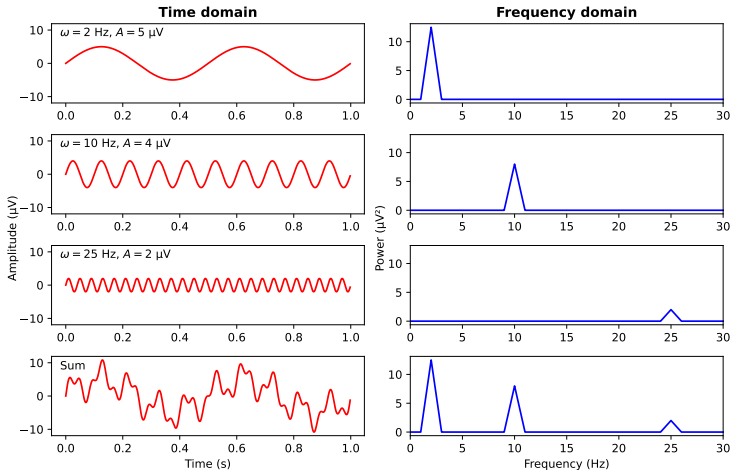
Joseph Fourier (1768–1830):

Any signal can be expressed as sum of weighted sine waves, each with its own frequency, amplitude, and phase

$$f(t) = A_0 + A_1 \cos(\omega t + \varphi_1) + A_2 \cos(\omega t + \varphi_2) + \dots + A_N \cos(\omega t + \varphi_N)$$

$$= \sum_{n=0}^N A_n \cos(\omega t + \varphi_n) \quad \text{where } A = \text{amplitude, } \omega = \text{frequency, } \varphi = \text{phase}$$

Fourier transform

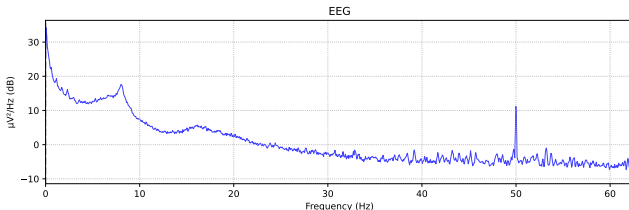


Frequency analysis in MNE-Python

```
# Read raw data
from mne.io import read_raw_brainvision
raw = read_raw_brainvision('data/raw/05.vhdr')

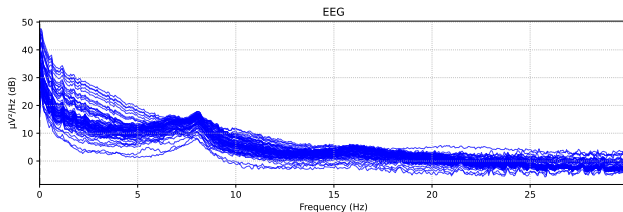
# Downsample to make subsequent computations faster
raw = raw.resample(125.)

# Plot spectrum for a single channel
_ = raw.plot_psd(picks='Cz', color='b', spatial_colors=False)
```

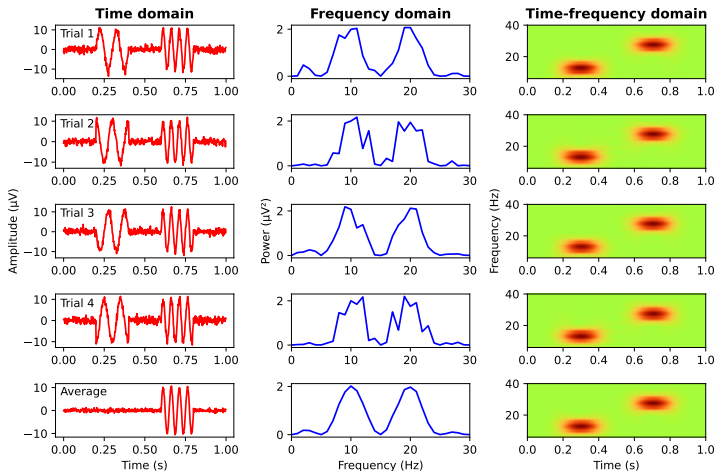


Frequency analysis in MNE-Python

```
# Plot spectrum for all channels, restricted to 0-30 Hz
_ = raw.plot_psd(fmax=30, color='b', spatial_colors=False)
```



Frequency analysis

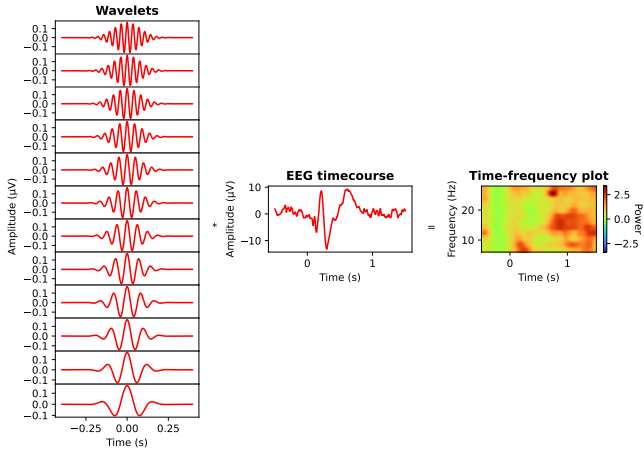


Time-frequency analysis

Goal: Estimate how power (or phase) at each frequency changes over time, e.g., in response to a stimulus

Approach: Apply the Fourier transform to a short time window and move this window over time points in the epoch → Short-term Fourier transform (STFT), Morlet wavelet convolution

Morlet wavelets



Baseline correction

- Should fit at least one full cycle at the lowest frequency
 - E.g., 5 Hz \rightarrow min. 200 ms baseline
- Should account for the $1/f$ scaling \rightarrow Divisive baseline
 - Subtract + divide by mean baseline: Percent signal change
 - Divide by mean baseline + take logarithm: Decibel
- Baseline should end *before* rather than *at* stimulus onset
 - Prevents post-stimulus activity “smearing” into the baseline

Time-frequency analysis in MNE-Python

```
# Load functions
import numpy as np
from mne import events_from_annotations, Epochs
from mne.time_frequency import tfr_morlet

# Segment continuous signal to epochs
events, _ = events_from_annotations(raw, regexp='Stimulus')
triggers = [201, 205]
epochs = Epochs(raw, events, triggers, tmin=-0.5, tmax=1.5, baseline=(-0.2, 0.))
print(epochs.get_data().shape) # Dimensions: (trials, channels, time points)

## (240, 64, 251)

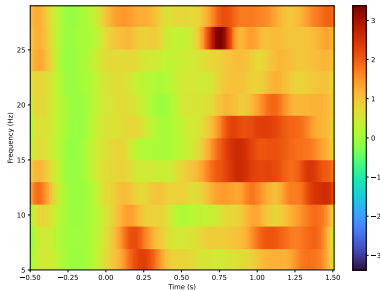
# Apply Morlet wavelet decomposition
freqs = np.arange(6., 30., step=2.)
n_cycles = np.arange(3., 15., step=1.)
tfr = tfr_morlet(epochs, freqs, n_cycles, return_itc=False, average=False)
print(tfr.data.shape) # Dimensions: (trials, channels, frequencies, time points)

## (240, 64, 12, 251)
```


Time-frequency analysis in MNE-Python

```
# Divisive baseline correction to get percent signal change
tfr = tfr.apply_baseline((-0.3, -0.1), mode='percent')

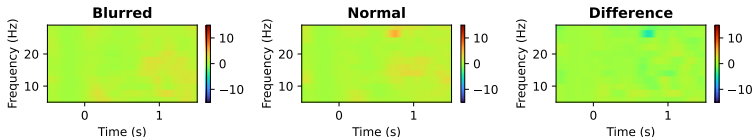
# Plot power at one channel, averaged across epochs
tfr_ave = tfr.average()
tfr_ave.plot(picks='Cz', cmap='turbo')
```



Time-frequency analysis in MNE-Python

```
# Compute power per condition and the difference between conditions
tfr_aves = {'Blurred': tfr['201'].average(),
            'Normal': tfr['205'].average(),
            'Difference': tfr['201'].average() - tfr['205'].average()}

# Create multi-panel plot
import matplotlib.pyplot as plt
fig, axs = plt.subplots(nrows=1, ncols=len(tfr_aves), figsize=(9, 2))
for ix, (label, tfr_ave) in enumerate(tfr_aves.items()):
    axs[ix].set_title(label, weight='bold')
    tfr_ave.plot(picks='Cz', cmap='turbo', vmin=-15, vmax=15,
                axes=axs[ix], show=False)
```



Time-frequency analysis with the pipeline

General settings:

```
# Import the Python package from R
pipeline <- reticulate::import("pipeline")

# Run the pipeline with the `tfr` (time-frequency) options
res <- pipeline$group_pipeline(
  vhdr_files = "data/raw",
  log_files = "data/log",
  output_dir = "output",
  ocular_correction = "data/cali",
  triggers = c(201:208, 211:218),
  average_by = c("n_b", "DeviantPosRL", "n_b/DeviantPosRL"),
  perform_tfr = TRUE,
  tfr_freqs = seq(6, 30, by = 2),
  tfr_cycles = seq(3, 15, by = 1),
  tfr_baseline_tmin = -0.3,
  tfr_baseline_tmax = -0.1,
  ... # See next 2 slides
)
```

Time-frequency analysis with the pipeline

For effects with *a priori* knowledge about their distribution:

```
# Define time-frequency "components" of interest
res <- pipeline$group_pipeline(
  ...,
  tfr_components = list(
    "name" = list("alpha"),
    "tmin" = list(0.0),
    "tmax" = list(0.2),
    "fmin" = list(8),
    "fmax" = list(14),
    "roi" = list(
      c("P09", "P07", "P03", "P0z", "P04", "P08", "P010", "01", "0z", "02")
    )
  )
)
```

→ Then use the single trial data frame to run mixed models

Time-frequency analysis with the pipeline

For exploratory analyses → Cluster-based permutation tests:

```
# Run 3D cluster-based permutation tests for contrasts of interest
res <- pipeline$group_pipeline(
  ...,
  perm_contrasts = list(
    c("blurr", "normal"),
    c("blurr/re", "blurr/li"),
    c("normal/re", "normal/li")
  )
)
```

→ Creates a new data frame with cluster-level p -values across time, space (channels), and frequencies

General remarks

- Richer view of the EEG signal (evoked + induced activity)
- Many new parameters to think carefully about:
 - Design → Longer inter-trial interval; jittering
 - Wavelet frequencies + number of cycles → T/f tradeoff
 - Baseline correction window and method
 - Interpretation:
 - Narrow-band vs. broad-band
 - Oscillation vs. rate of change
- Everything matters

Thanks

