Turning MNE-based analysis into an interactive app

Nikolai Kapralov
MPI CBS Leipzig

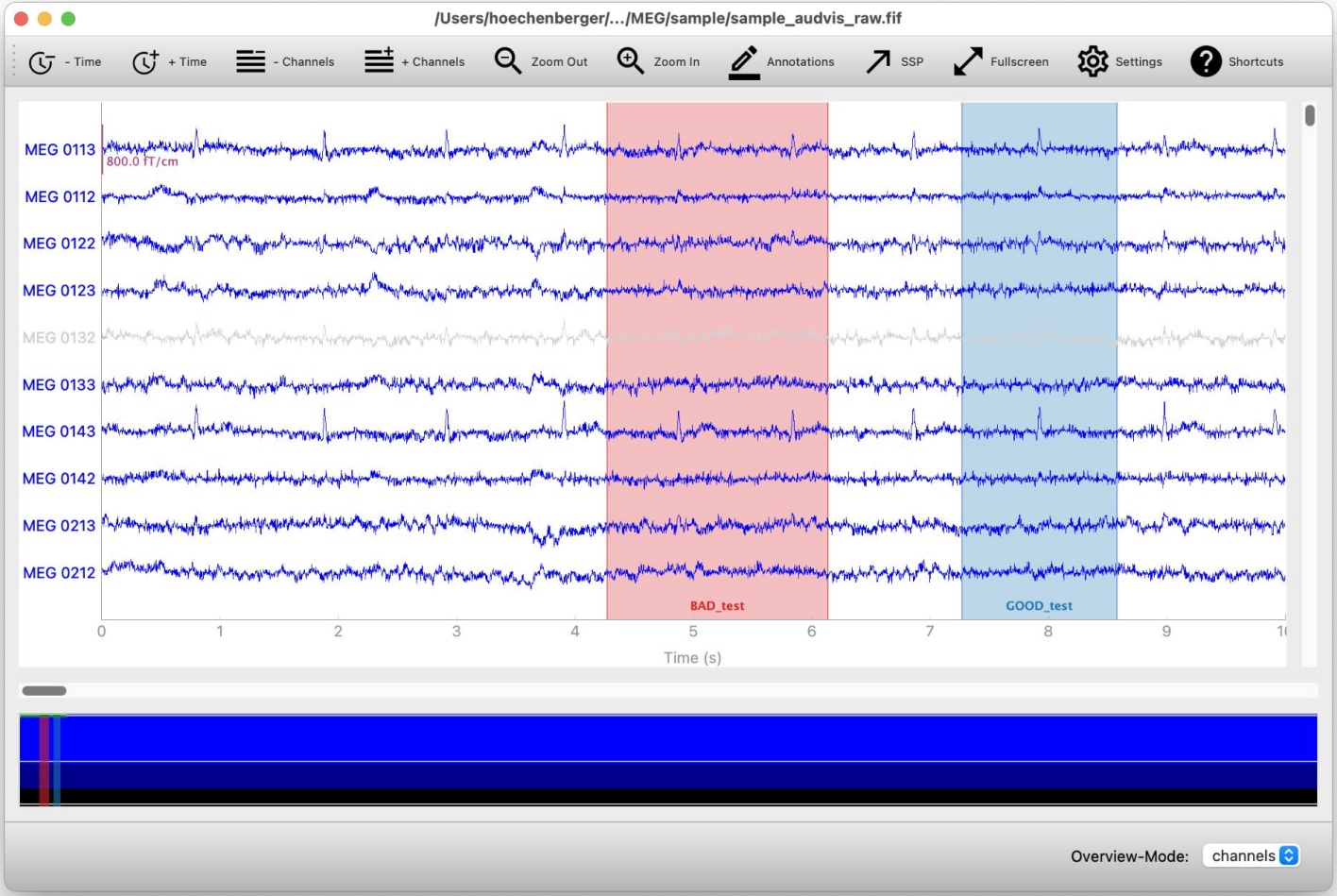
Berlin Brainwaves, 12-13.10.2023

Why?

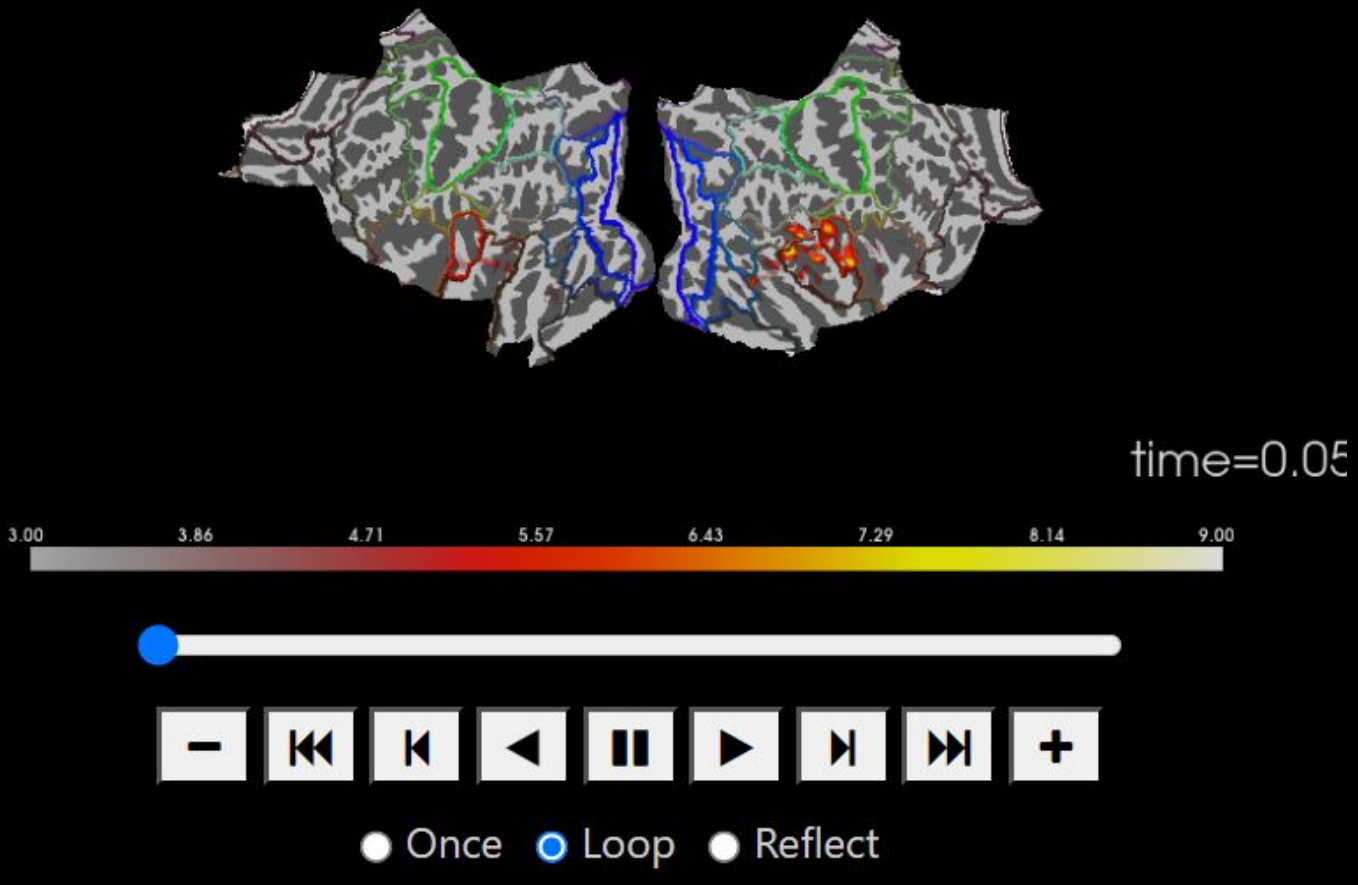
Why?

- Convenience

raw.plot() -> mne-qt-browser



mne.SourceEstimate.plot() -> mne.viz.Brain



Why?

- Convenience
  - MNE-Python already provides several interactive apps

Why?

- Convenience
  - MNE-Python already provides several interactive apps
- Deeper understanding

Explorable Multiverse Analysis

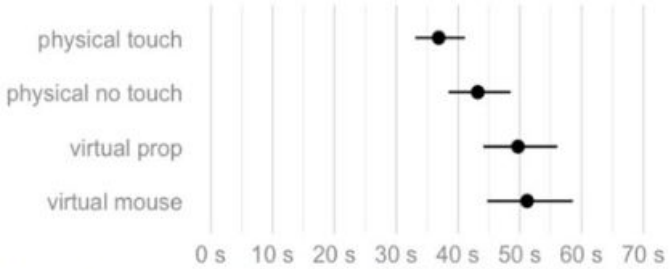
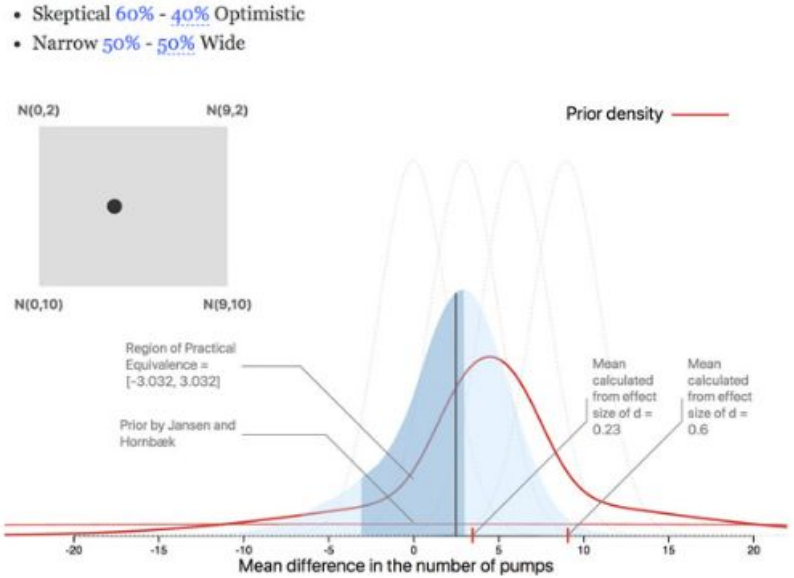


Figure 3. Average task completion time (geometric mean) for each condition. Error bars are 95% t-based CIs.

We focus our analysis on task completion times, reported in Figures 3 and 4. Dots indicate sample means, while error bars are 95% confidence intervals computed on log-transformed data [6] using the t-distribution method. Strictly speaking, all we can assert about each interval is that it comes from a procedure designed to capture the



read more here and there

Why?

- Convenience
  - MNE-Python already provides several interactive apps
- Deeper understanding
  - Interactively explore the underlying data and methods: explorable multiverse analysis

raw.compute_psd()

```
compute_psd(method='welch', fmin=0, fmax=inf, tmin=None, tmax=None,
picks=None, proj=False, reject_by_annotation=True, *, n_jobs=1,
verbose=None, **method_kw)                                    [source]
```

****method_kw**

Additional keyword arguments passed to the spectral estimation function (e.g., `n_fft, n_overlap, n_per_seg, average, window` for Welch method, or `bandwidth, adaptive, low_bias, normalization` for multitaper method). See `psd_array_welch()` and `psd_array_multitaper()` for details.

Why?

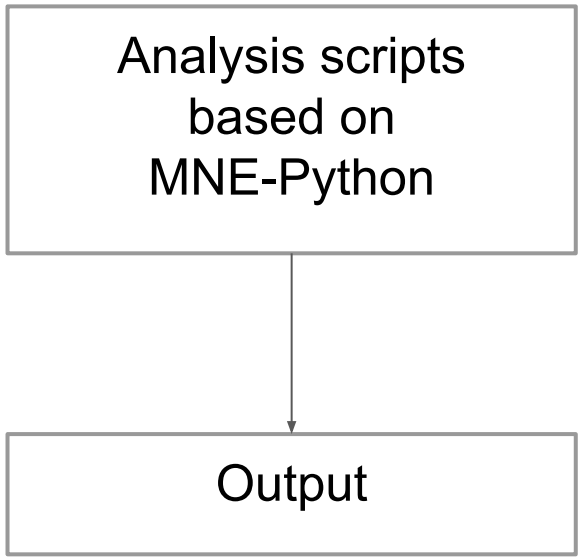
- Convenience
  - MNE-Python already provides several interactive apps
- Deeper understanding
  - Interactively explore the underlying data and methods: explorable multiverse analysis
  - Make it easier and more satisfying to try out different parameters (also when developing methods)

Why?

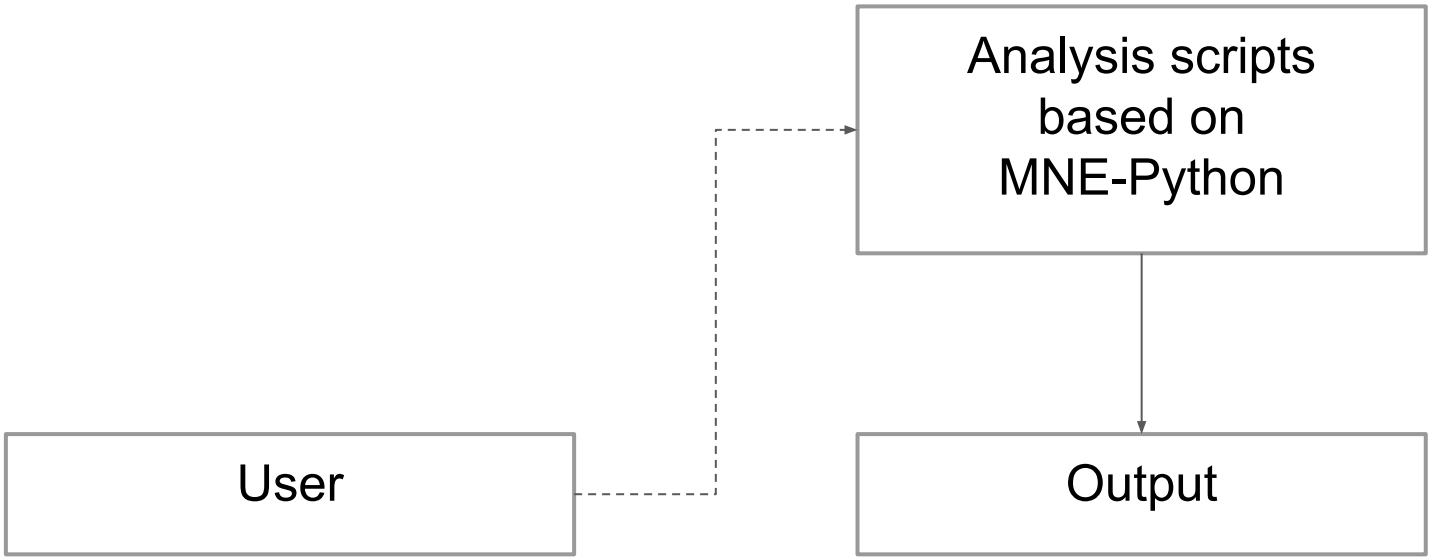
- Convenience
  - MNE-Python already provides several interactive apps
- Deeper understanding
  - Interactively explore the underlying data and methods: explorable multiverse analysis
  - Make it easier and more satisfying to try out different parameters (also when developing methods)
- For fun

How?

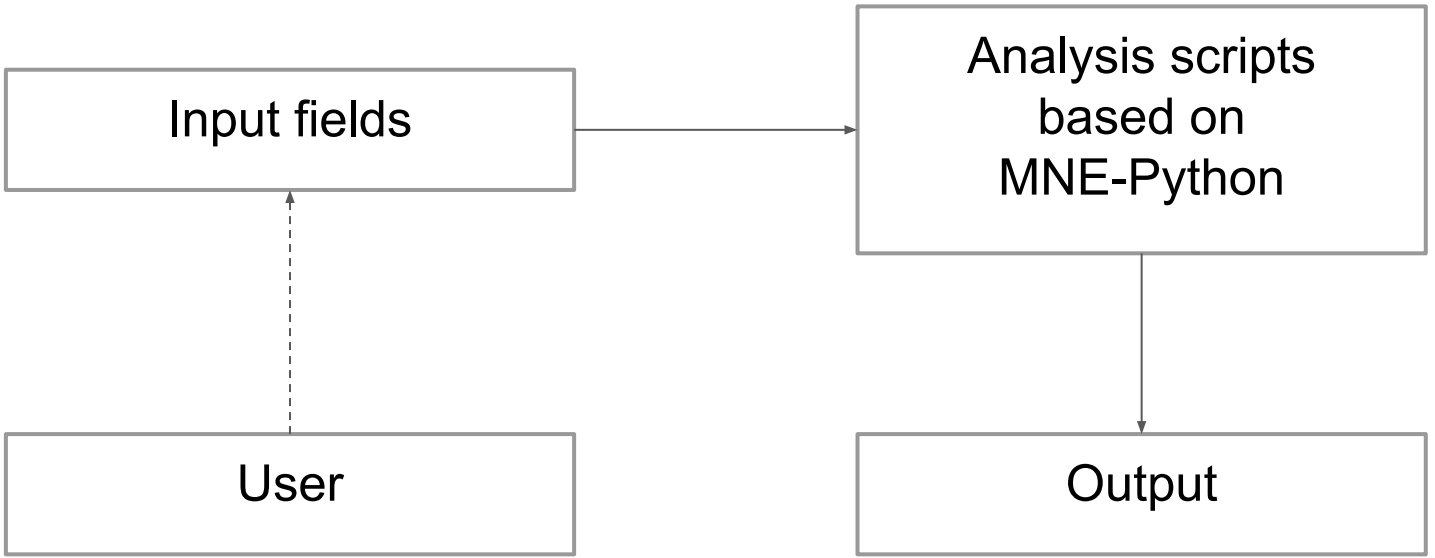
How?



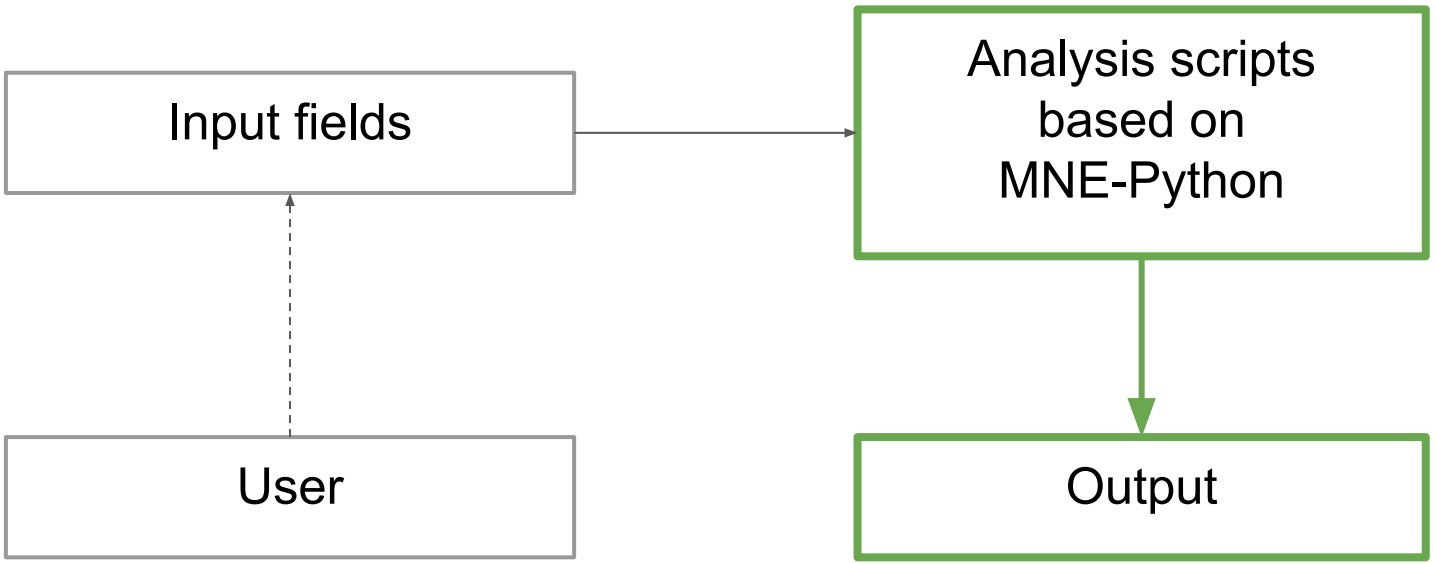
How?



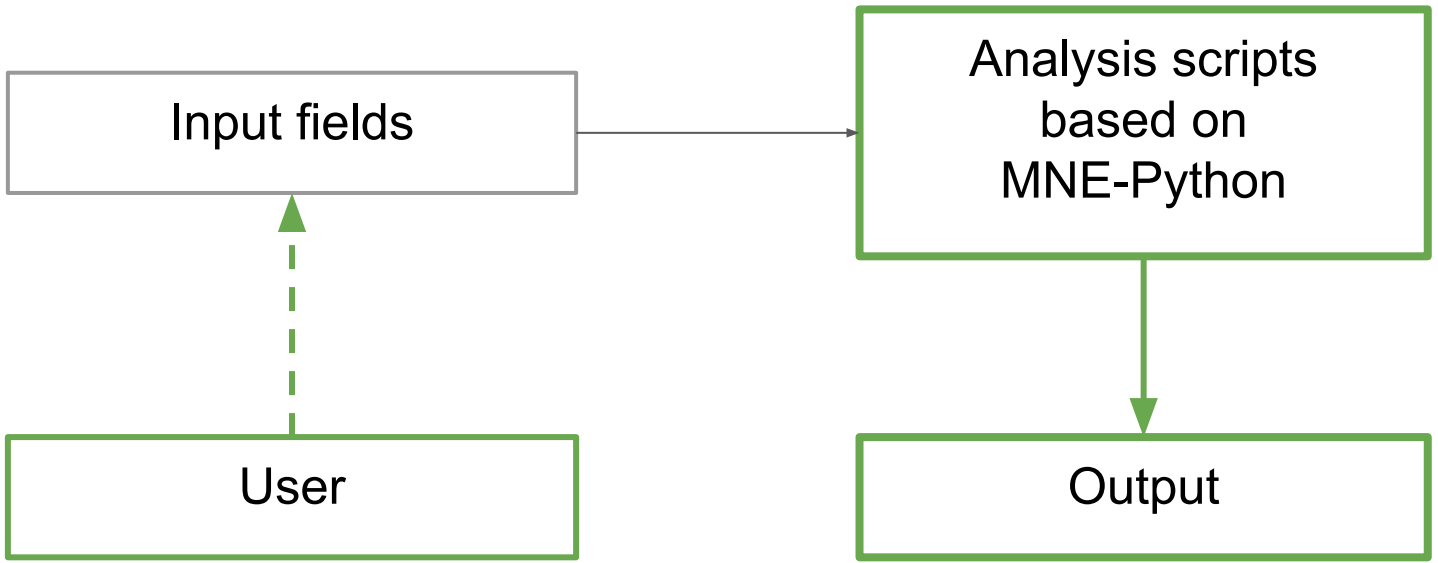
How?


Input fields
Analysis scripts based on MNE-Python
User
Output

How?


Input fields
Analysis scripts based on MNE-Python
User
Output

How?


Input fields
Analysis scripts based on MNE-Python
User
Output

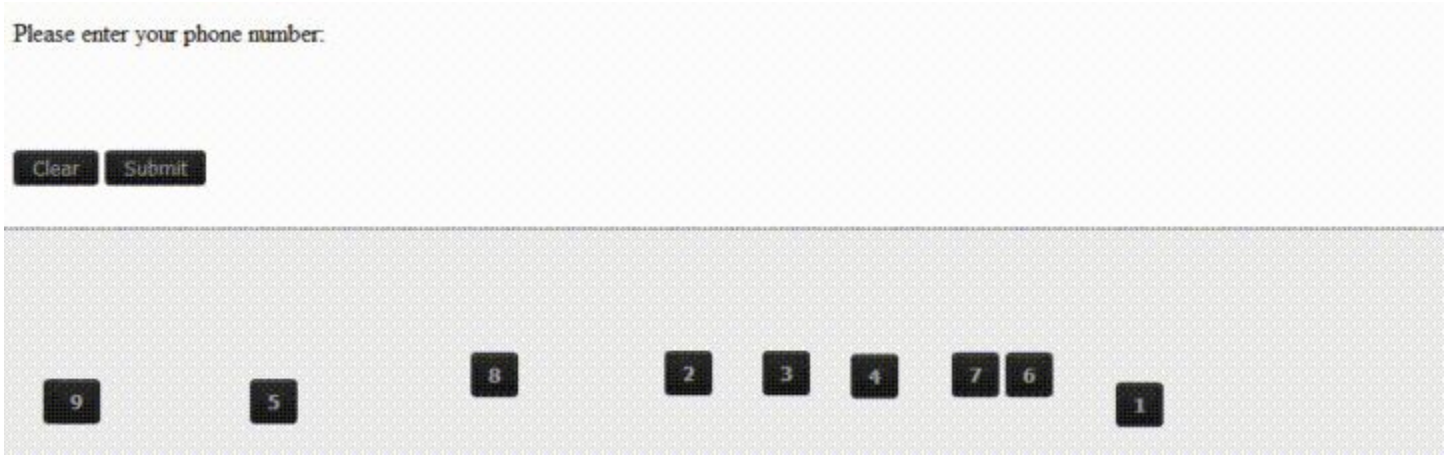
The powerful data exploration & web app framework for Python

Streamlit

A faster way to build and share data apps

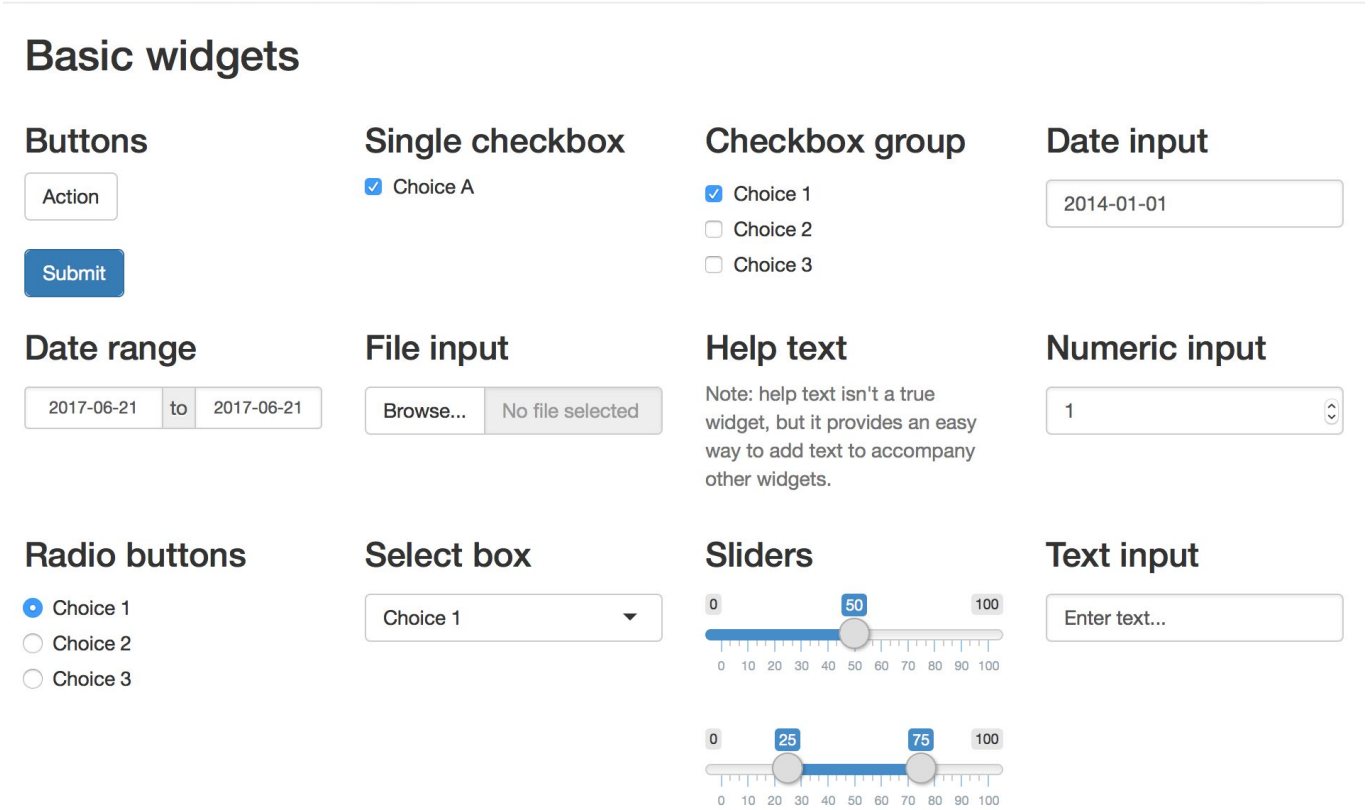
Streamlit turns data scripts into shareable web apps in minutes.
All in pure Python. No front-end experience required.

Input Fields / Widgets

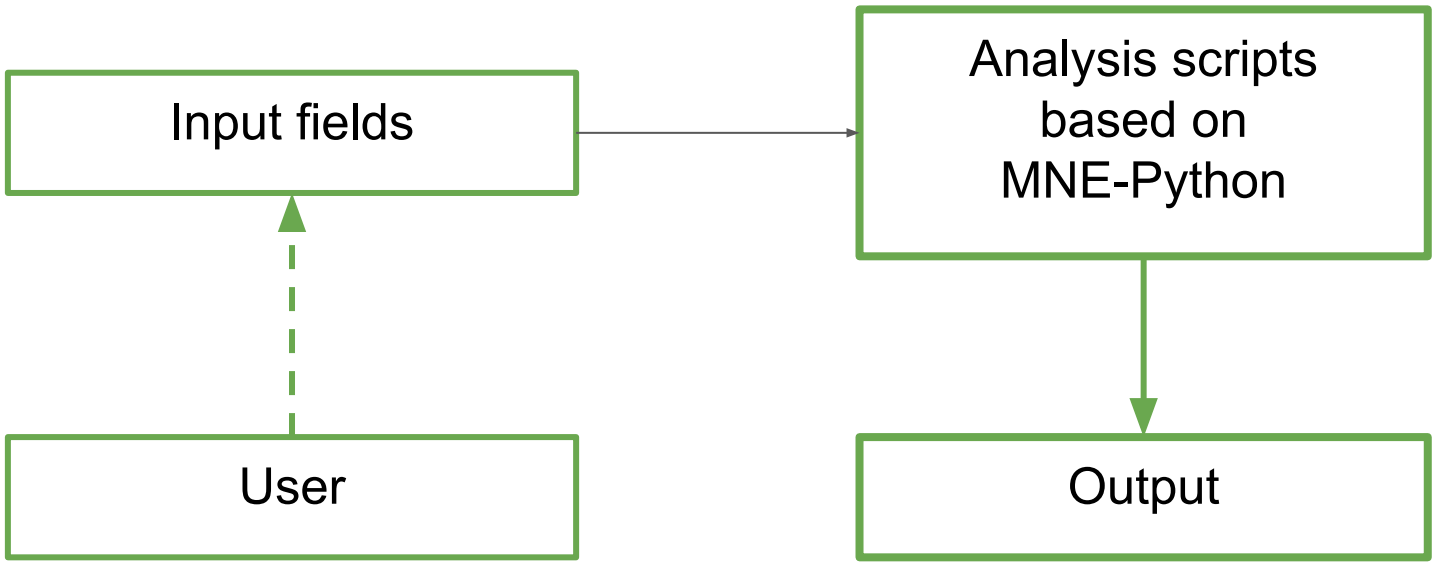


more examples

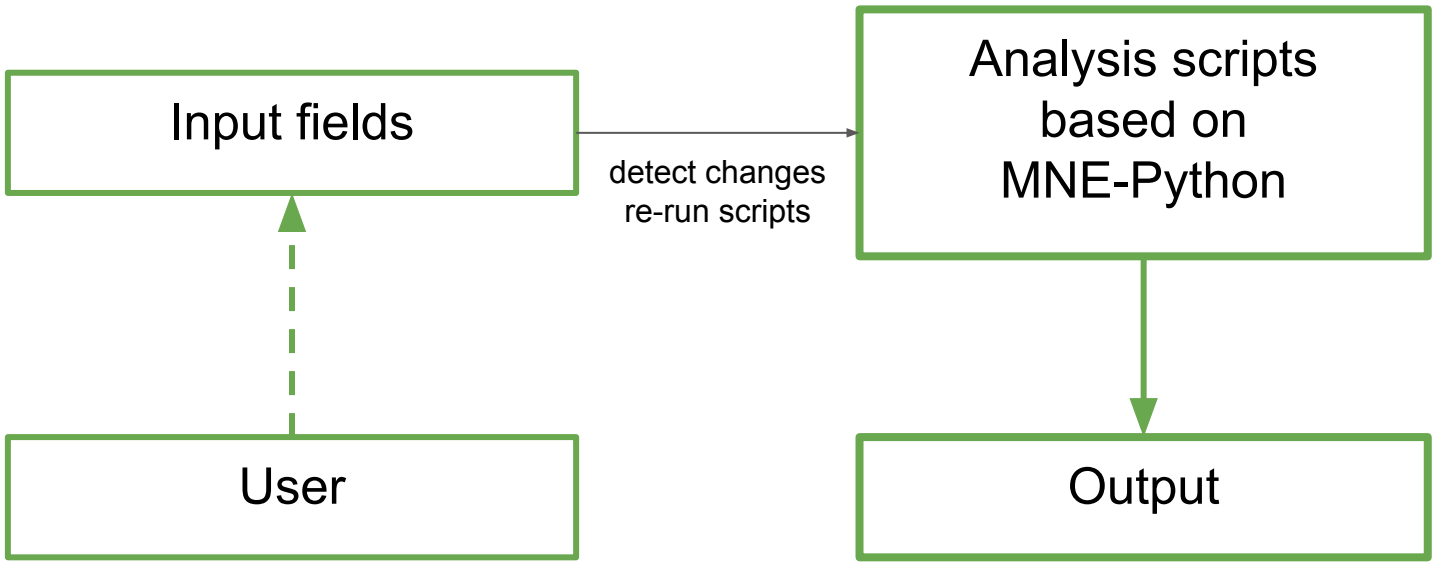
Input Widgets



How?

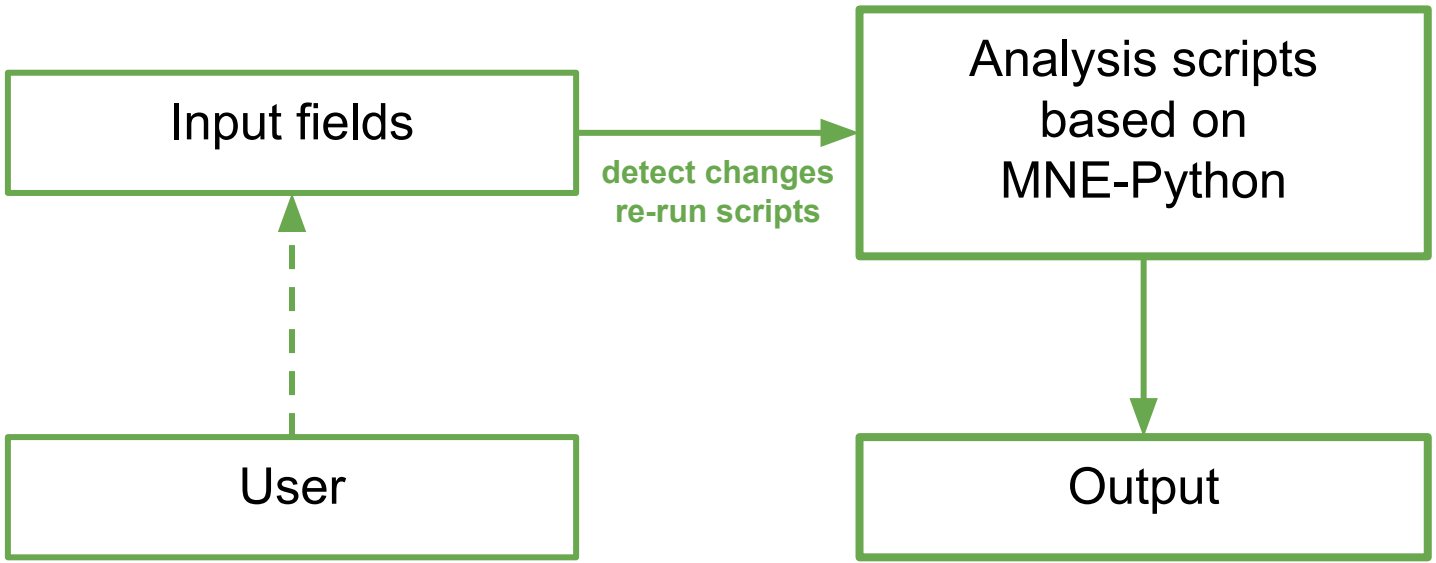

Input fields
Analysis scripts based on MNE-Python
User
Output

How?


Input fields
detect changes
re-run scripts
Analysis scripts
based on
MNE-Python
User
Output

Result

Web Application



Any questions?

Practical part

- Panel (`panel_spectrum.ipynb`)
  - Minimal example to try out widgets
  - Parameters of methods for estimation of PSD in MNE-Python
- Streamlit
  - Minimal example (`streamlit_minimal.py`)
  - Parameters of FOOOF for splitting the PSD into aperiodic and periodic components (`streamlit_fooof.py`)

Practical part

- Panel (`panel_spectrum.ipynb`)
  - Minimal example to try out widgets
  - Parameters of methods for estimation of PSD in MNE-Python
- Streamlit
  - Minimal example (`streamlit_minimal.py`)
  - Parameters of FOOOF for splitting the PSD into aperiodic and periodic components (`streamlit_fooof.py`)

➢ Tutorials *should* be suitable for self-paced mode

Practical part

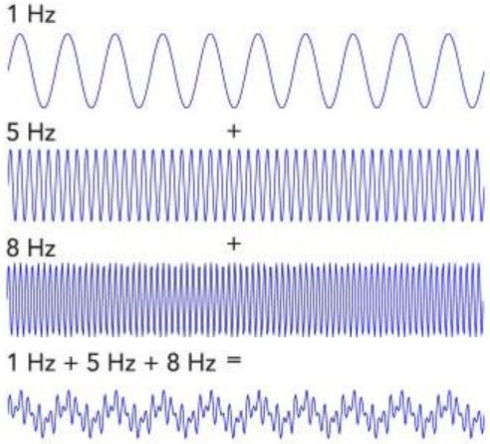
- Panel (`panel_spectrum.ipynb`)
  - Minimal example to try out widgets
  - Parameters of methods for estimation of PSD in MNE-Python
- Streamlit
  - Minimal example (`streamlit_minimal.py`)
  - Parameters of FOOOF for splitting the PSD into aperiodic and periodic components (`streamlit_fooof.py`)

➢ Tutorials *should* be suitable for self-paced mode

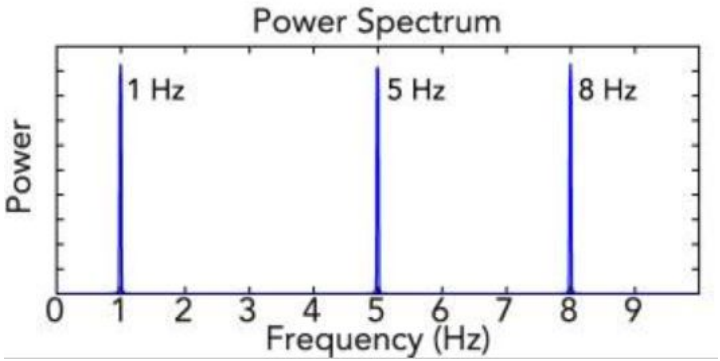
➢ Folder with solutions is provided

Estimation of PSD - Fourier Transform

Prerau et al. (2017)



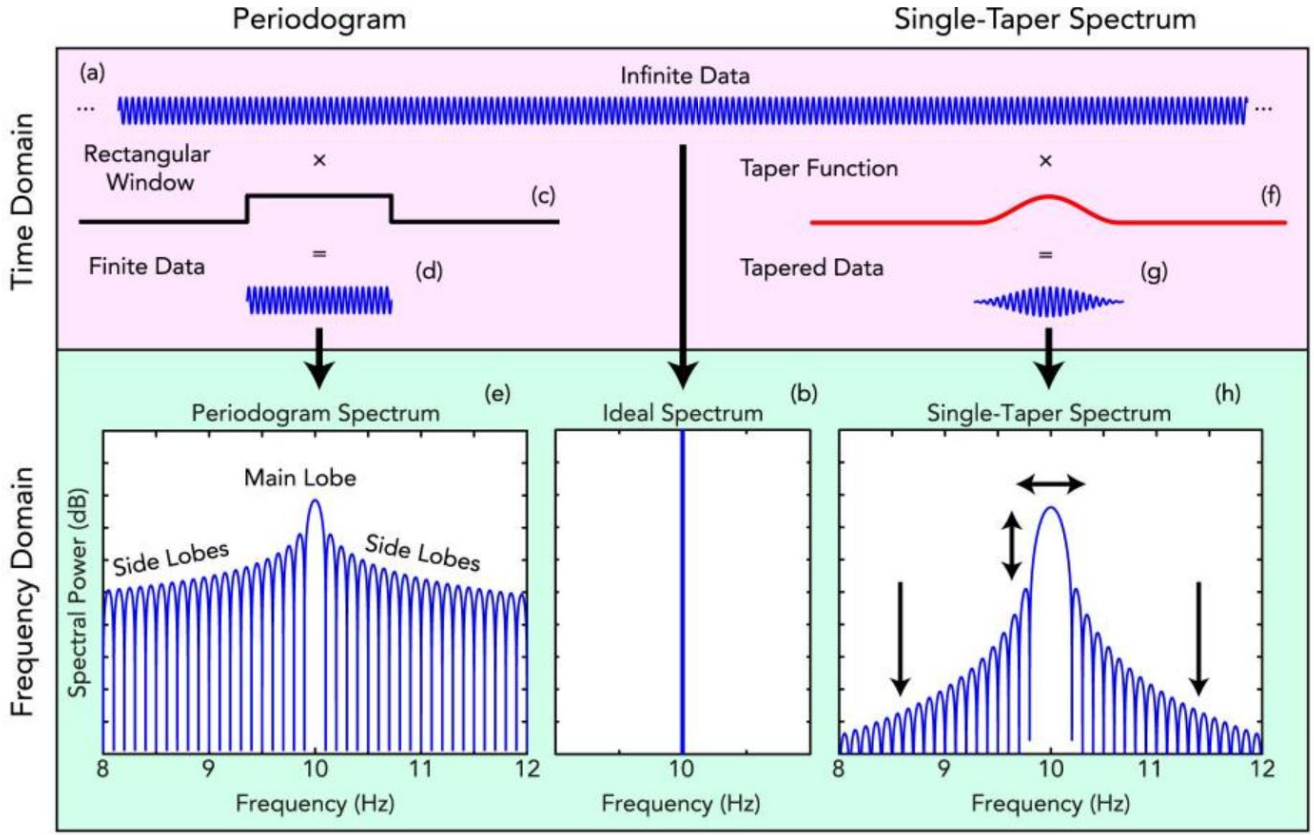
Time Domain



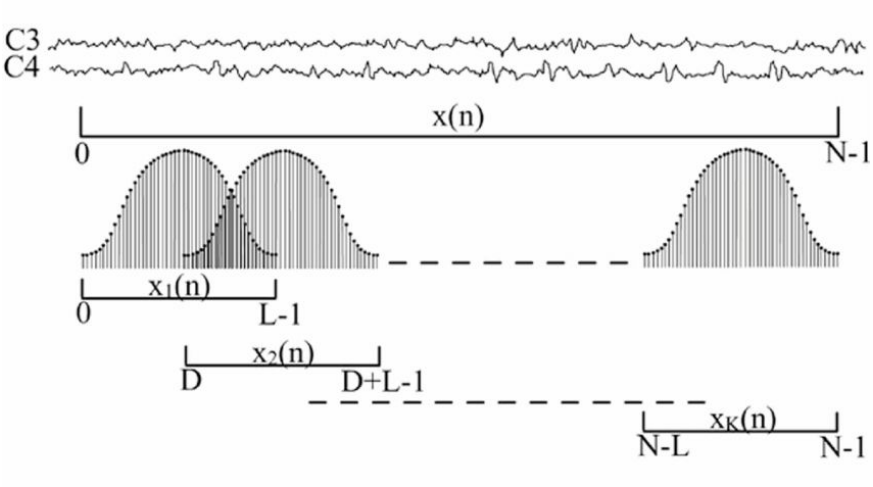
Frequency Domain

Estimation of PSD - Spectral Leakage

Prerau et al. (2017)



Estimation of PSD - Welch Parameters



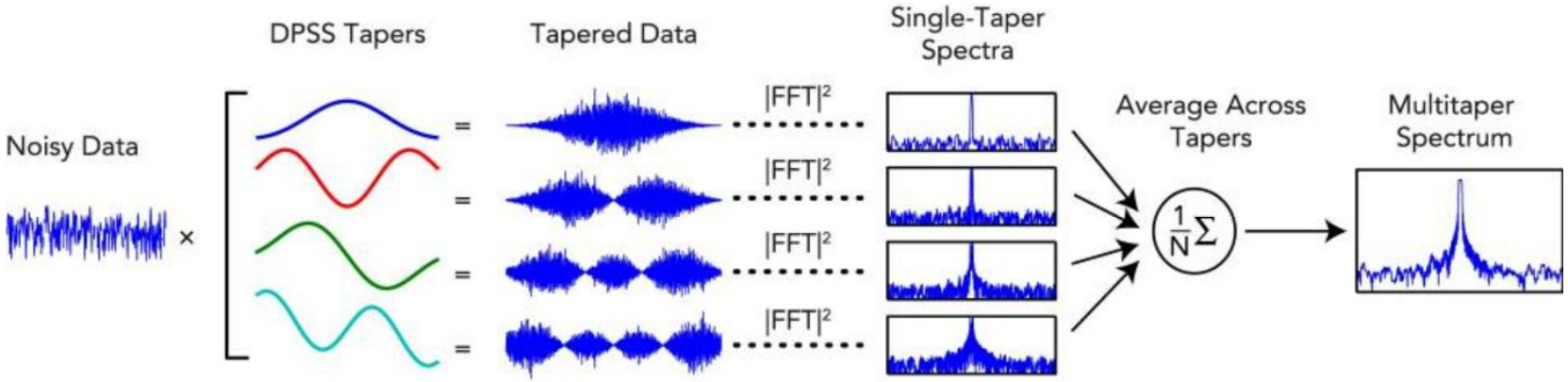
Welch's method:

- Split the data into overlapping segments
- Apply a window to each segment
- Calculate PSD on each segment using FFT and average

mne.time_frequency.psd_array_welch()

```
mne.time_frequency.psd_array_welch(x, sfreq, fmin=0, fmax=inf,
n_fft=256, n_overlap=0, n_per_seg=None, n_jobs=None, average='mean',
window='hamming', *, output='power', verbose=None)                [source]
```

Estimation of PSD - Multitaper Parameters



mne.time_frequency.psd_array_multitaper()

```
mne.time_frequency.psd_array_multitaper(x, sfreq, fmin=0.0, fmax=inf,
bandwidth=None, adaptive=False, low_bias=True, normalization='length',
output='power', n_jobs=None, *, max_iter=150, verbose=None)
```

[source]

Panel

PSD: Welch vs Multitaper

Common:

Channel: EEG028

Time Range (s): **0 .. 30**

Frequency Range (Hz): **0 .. 100**

Welch:

n_fft: **800**

n_overlap: **400**

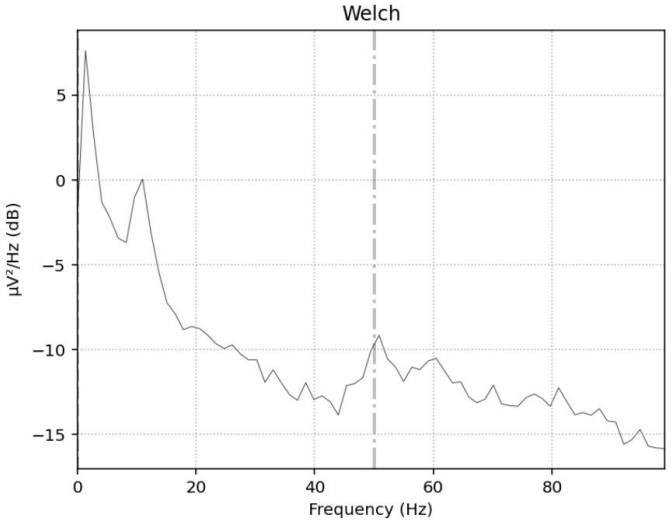
Window: hamming

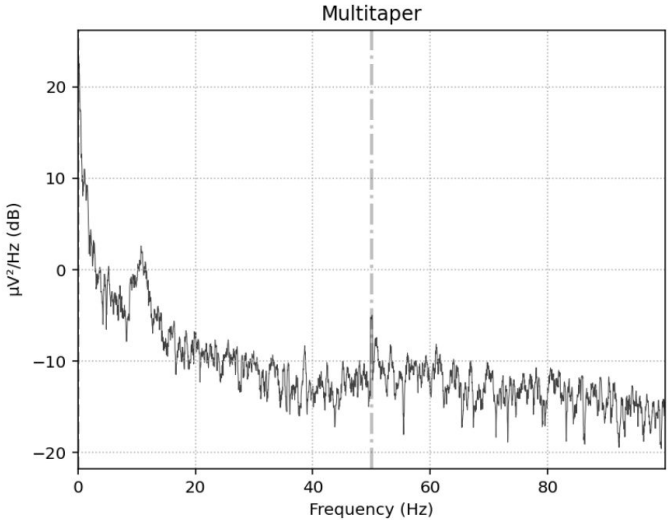
Multitaper:

Normalization: full

Half-Bandwidth: **5**

Low bias

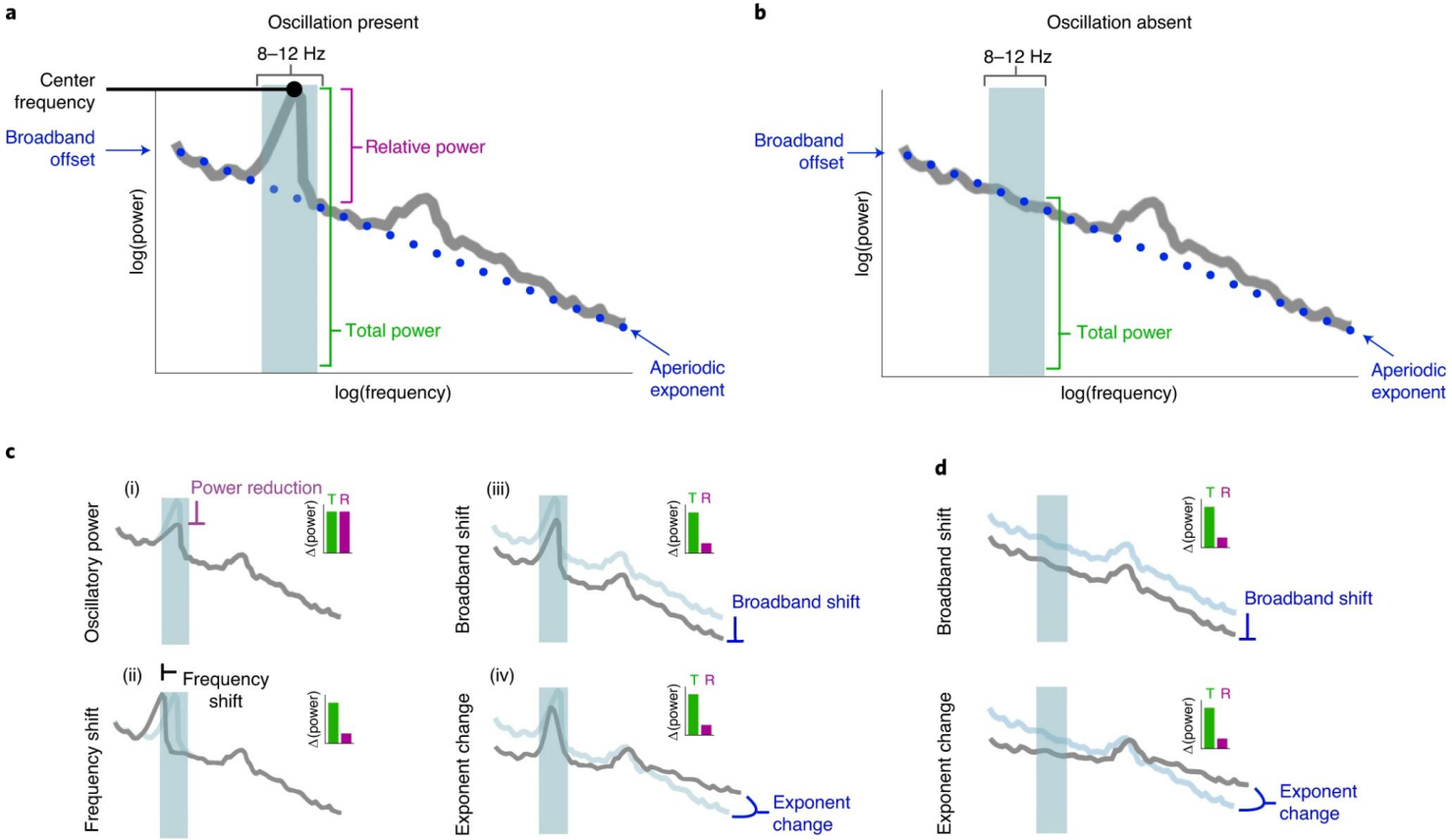




Any questions?

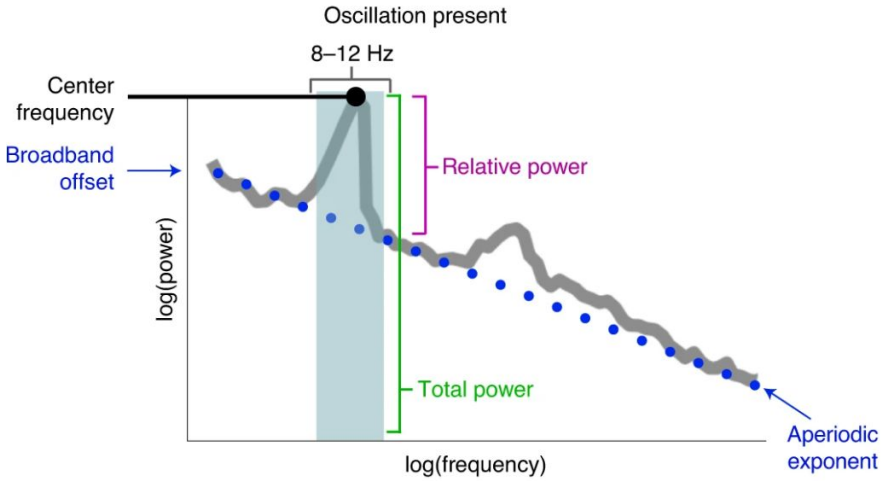
Donoghue et al. (2020)

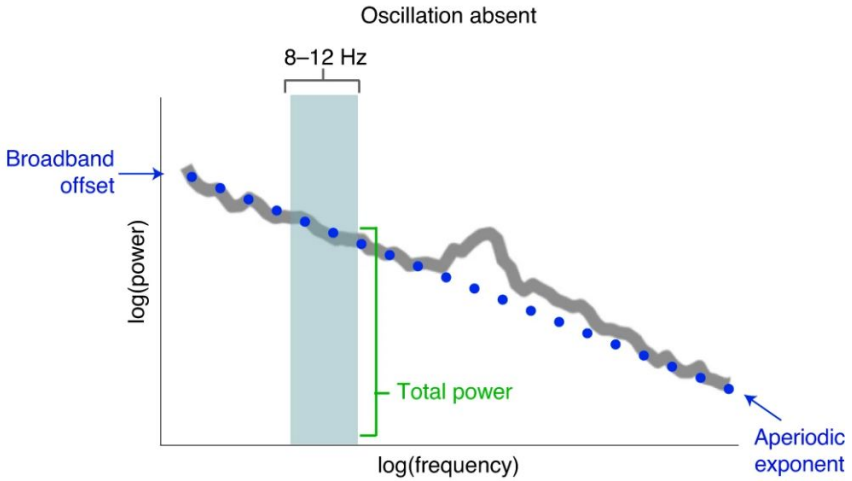
FOOOF = Fitting Oscillations and One-Over-F



Donoghue et al. (2020)

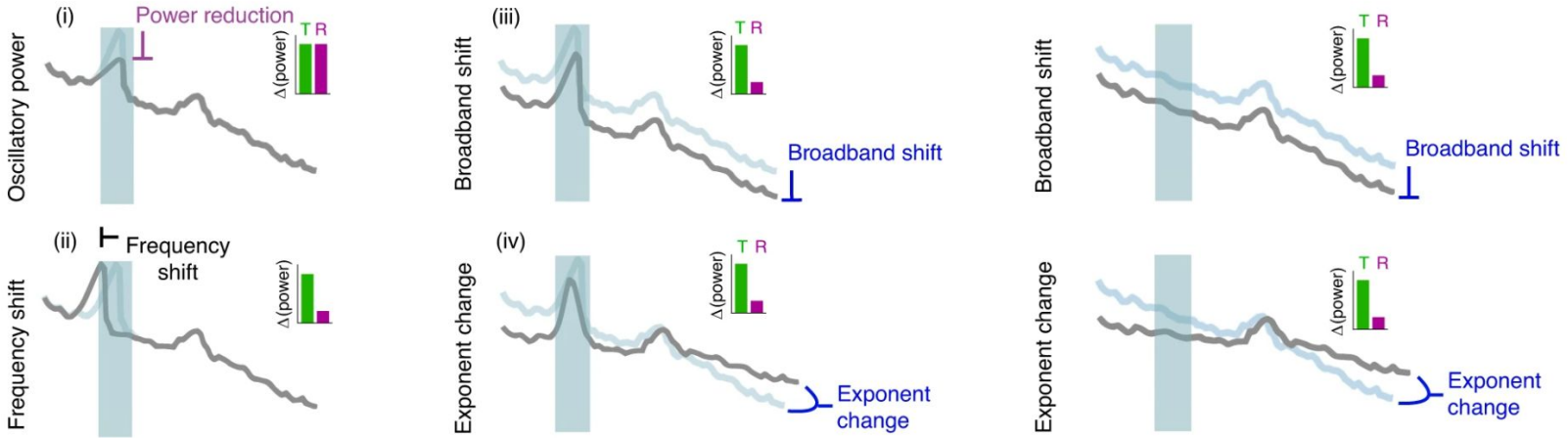
FOOOF = Fitting Oscillations and One-Over-F





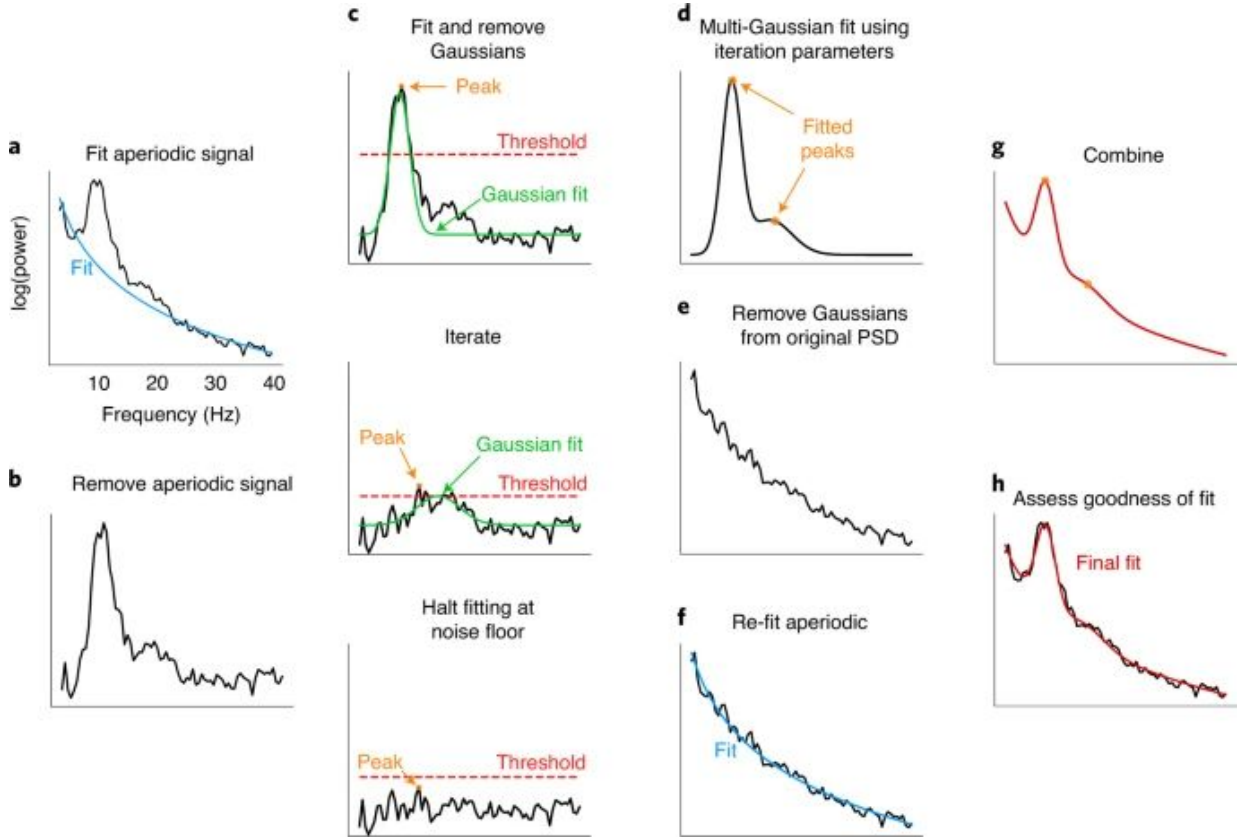
Donoghue et al. (2020)

FOOOF = Fitting Oscillations and One-Over-F



Donoghue et al. (2020)

FOOOF = Fitting Oscillations and One-Over-F



FOOOF = Fitting Oscillations and One-Over-F

```
fooof.FOOOF()
```

```
class fooof.FOOOF(peak_width_limits=(0.5, 12.0),
max_n_peaks=inf, min_peak_height=0.0, peak_threshold=2.0,
aperiodic_mode='fixed', verbose=True)                    [source]
```

```
fooof.FOOOF.fit()
```

```
fit(freqs=None, power_spectrum=None, freq_range=None)    [source]
```

Streamlit

Channel

EEG039

Frequency range (Hz)

1 54

0 250

Peak width limits (Hz)

2.00 18.00

0.00 40.00

Maximal number of peaks

3

1 10

Aperiodic mode

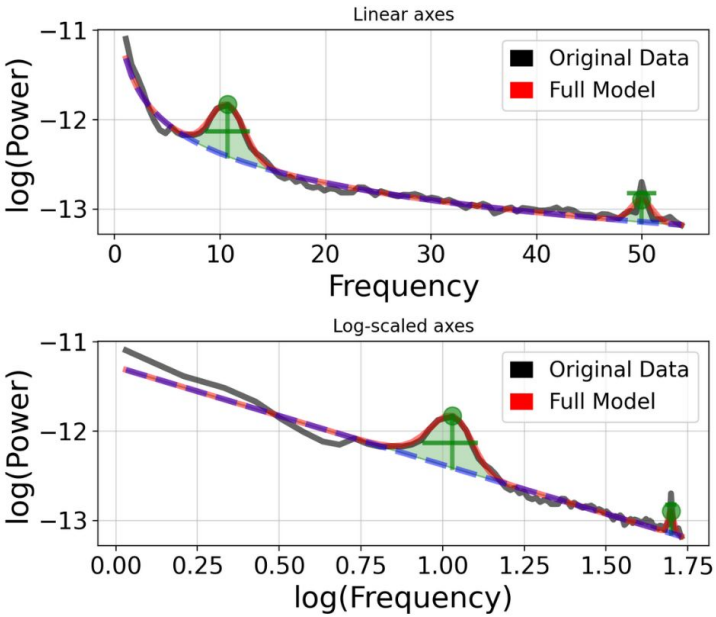
fixed knee

Annotate peaks

Annotate aperiodic

FOOOF

Expand to see more details about FOOOF!



t

-11.28

Exponent

1.1

Peak frequency (Hz)
[first peak only]

10.71

Thank you for attention!

Questions/suggestions: kapralov@cbs.mpg.de