fooof_plotting_PR

November 17, 2022

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np
import scipy.signal as sig
from fooof import FOOOF
from fooof.sim.gen import gen_aperiodic
from numpy.fft import irfft, rfftfreq
import scipy as sp
import scipy.signal as sig
```

[ ]:
```
def elec_phys_signal(exponent: float,
                     periodic_params: list[tuple[float, float, float]] = None,
                     nlv: float = None,
                     highpass: bool = False,
                     sample_rate: float = 2400,
                     duration: float = 180,
                     seed: int = 1):
    """
    Generate 1/f noise with optionally added oscillations.
    Parameters
    ----------
    exponent : float
        Aperiodic 1/f exponent.
    periodic_params : list of tuples
        Oscillations parameters as list of tuples in form of
                [(center_frequency1, peak_amplitude1, peak_width1),
                (center_frequency2, peak_amplitude2, peak_width2)]
        for two oscillations.
    nlv : float, optional
        Level of white noise. The default is None.
    highpass : bool, optional
        Whether to apply a 4th order butterworth highpass filter at 1Hz.
        The default is False.
    sample_rate : float, optional
        Sample rate of the signal. The default is 2400Hz.
    duration : float, optional
        Duration of the signal in seconds. The default is 180s.
    seed : int, optional
```

```
    Seed for reproducability. The default is 1.
Returns
-------
aperiodic_signal : ndarray
    Aperiodic 1/f activitiy without oscillations.
full_signal : ndarray
    Aperiodic 1/f activitiy with added oscillations.
"""
if seed:
    np.random.seed(seed)
# Initialize
n_samples = int(duration * sample_rate)
amps = np.ones(n_samples//2, complex)
freqs = rfftfreq(n_samples, d=1/sample_rate)
freqs = freqs[1:]  # avoid divison by 0

# Create random phases
rand_dist = np.random.uniform(0, 2*np.pi, size=amps.shape)
rand_phases = np.exp(1j * rand_dist)

# Multiply phases to amplitudes and create power law
amps *= rand_phases
amps /= freqs ** (exponent / 2)

# Add oscillations
amps_osc = amps.copy()
if periodic_params:
    for osc_params in periodic_params:
        freq_osc, amp_osc, width = osc_params
        amp_dist = sp.stats.norm(freq_osc, width).pdf(freqs)
        # add same random phases
        amp_dist = amp_dist * rand_phases
        amps_osc += amp_osc * amp_dist

# Create colored noise time series from amplitudes
aperiodic_signal = irfft(amps)
full_signal = irfft(amps_osc)

# Add white noise
if nlv:
    w_noise = np.random.normal(scale=nlv, size=n_samples-2)
    aperiodic_signal += w_noise
    full_signal += w_noise

# Highpass filter
if highpass:
    sos = sig.butter(4, 1, btype="hp", fs=sample_rate, output='sos')
```

```
        aperiodic_signal = sig.sosfilt(sos, aperiodic_signal)
        full_signal = sig.sosfilt(sos, full_signal)

    return aperiodic_signal, full_signal
```

1 Simulate signal

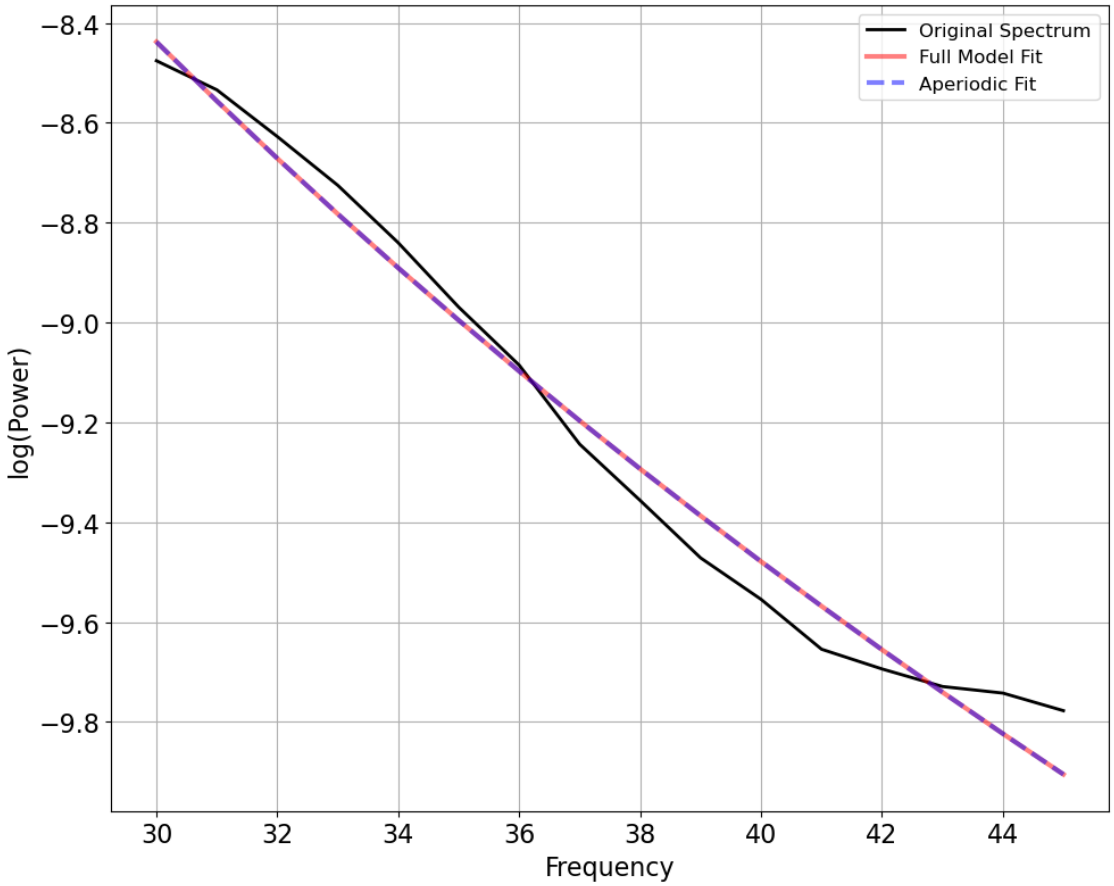
[ ]:
```
exponent = 1.5  # 1/f exponent
nlv = 0.0003  # white noise level

# Oscillations as tuple (frequency, amplitude, width)
med_delta = (2, 6, 1.9)
alpha = (12, 1.7, 3)
low_beta = (18, 2, 2)
high_beta = (27, 20, 6)
gamma = (50, 6, 15)
power_line = (50, .025, .001)
hfo = (360, 20, 60)

oscillations = (med_delta, alpha, low_beta, high_beta, gamma, power_line, hfo)

# Make signal
_, full_med = elec_phys_signal(periodic_params=oscillations, exponent=exponent,␣
 ↪nlv=nlv, highpass=True)
```

2 Calc PSD and FOOOF

[ ]:
```
# Welch params
sample_rate = 2400
nperseg = sample_rate
welch_params = dict(fs=sample_rate, nperseg=nperseg)

# Calc PSD
freq, psd_med = sig.welch(full_med, **welch_params)

# Fit simulated spectrum
fm_med = FOOOF(max_n_peaks=0, verbose=False)
fm_med.fit(freq, psd_med, (30, 45))

ap_fit_med = gen_aperiodic(fm_med.freqs, fm_med.aperiodic_params_)
```

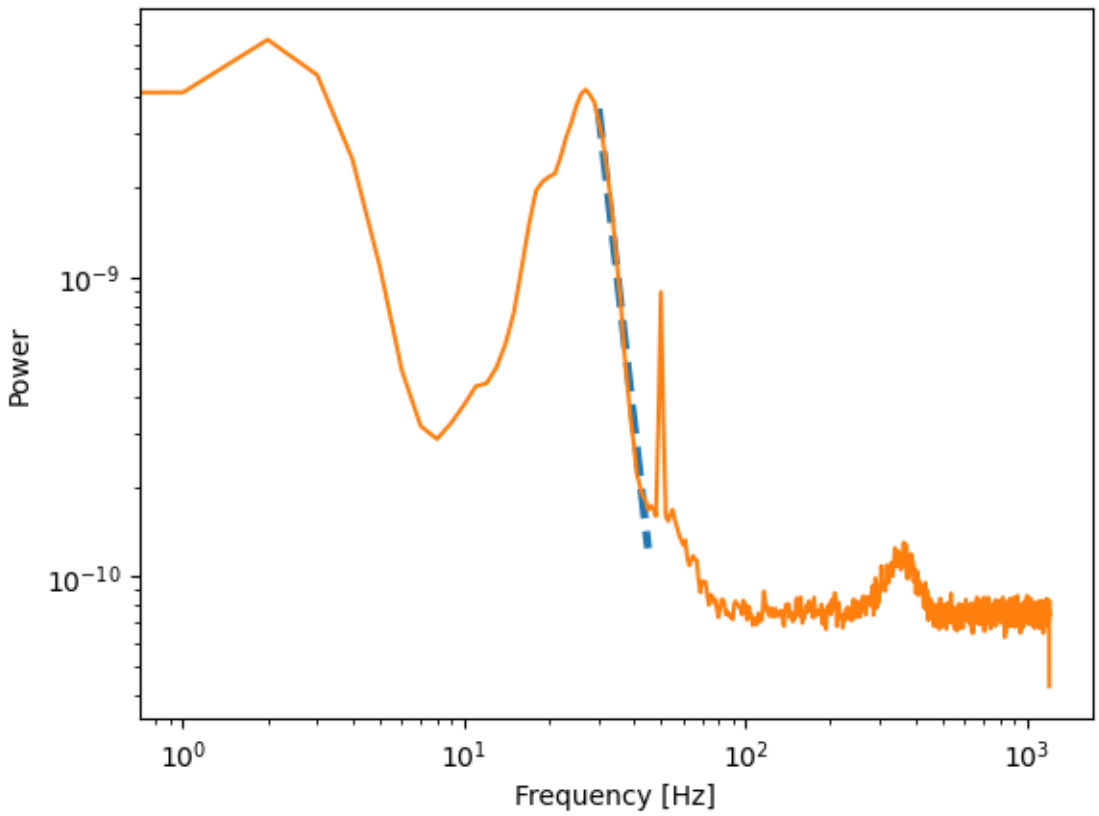
3 The FOOOF Plot looks fine...

[ ]:
```
fm_med.plot(plot_range=None)
```



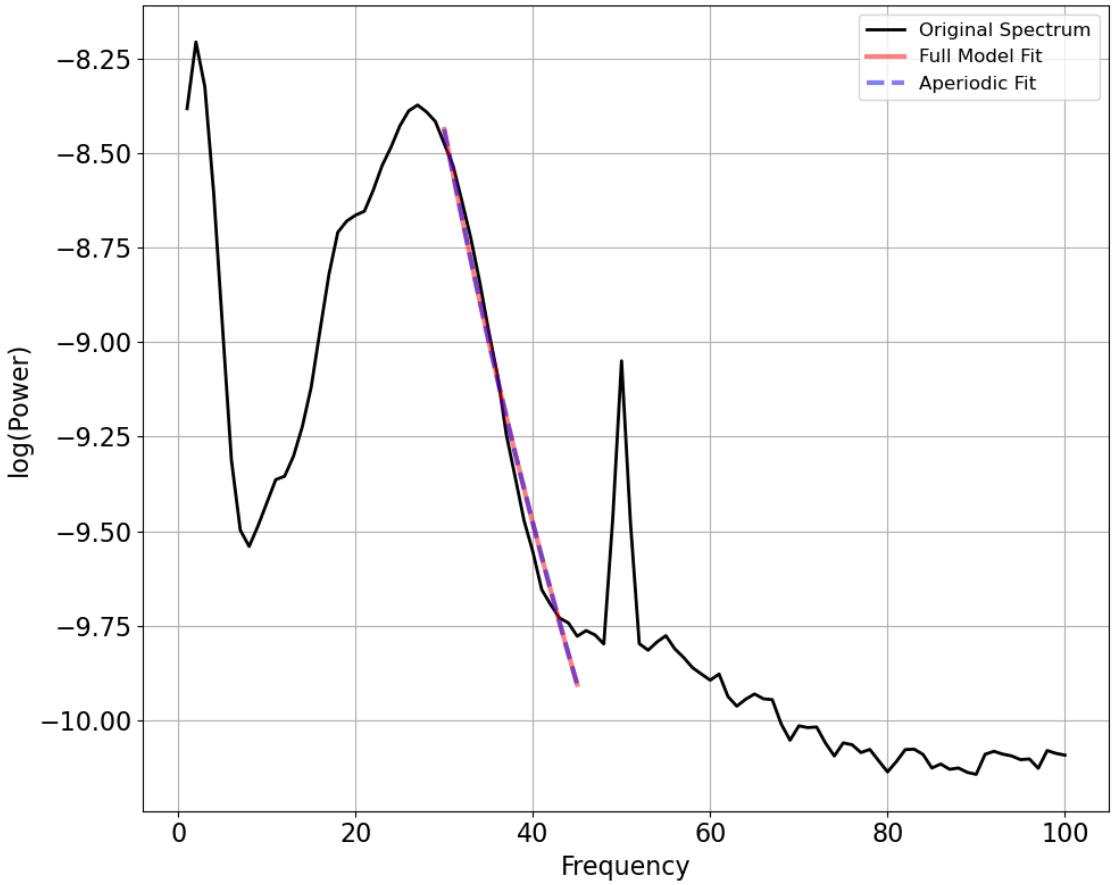
4 The full plot shows how bad it really is...

[ ]:
```
plt.loglog(fm_med.freqs, 10**ap_fit_med, "--", lw=3)
plt.loglog(freq, psd_med)
plt.xlabel("Frequency [Hz]")
plt.ylabel("Power")
plt.show()
```



5 New FOOOF Plot with modified code from this PR:

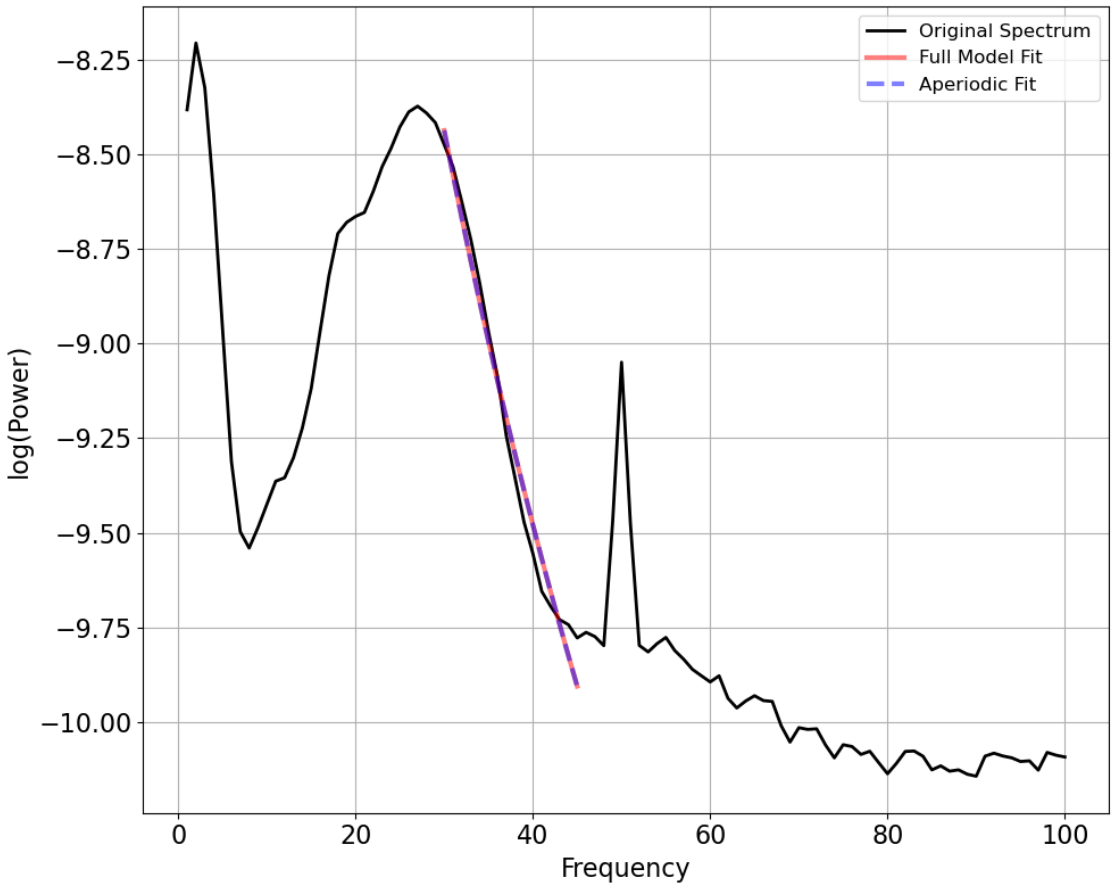
[ ]:
```
fm_med.plot(plot_range=(1, 100))
```



[ ]:
```
fm_med.report(plot_range=(1, 100))
```

```
================================================================================
==================

                                FOOOF - POWER SPECTRUM MODEL

                The model was run on the frequency range 30 - 45 Hz
                     Frequency Resolution is 1.00 Hz

                  Aperiodic Parameters (offset, exponent):
                              3.8718, 8.3334

                              0 peaks were found:

                            Goodness of fit metrics:
                           R^2 of model fit is 0.9813
                           Error of the fit is 0.0541
```

==========
==========


Original Spectrum
Full Model Fit
Aperiodic Fit
−8.25
−8.50
−8.75
−9.00
−9.25
−9.50
−9.75
−10.00
log(Power)
0
20
40
60
80
100
Frequency