MNE-Python:
Improve Time Frequency Analysis in Source Space

Sub-Organization Information: MNE-Python

Project Abstract:

Enable full application of Time-Frequency Analysis tools on Source Estimate M/EEG neurophysiological data by integrating `mne.SourceEstimate` objects with `mne.time_frequency.tfr_*` functions.

Student Information:

Name:	Dirk Gütlin
Github:	DiGyt
University:	University of Salzburg

Motivation:

Time-Frequency Analyses of induced neural oscillations pose an important aspect of neuroscience. While often measuring oscillations at the sensor level, spatial precision and overall information gain can be considerably enhanced by analyzing induced activity after projecting the measured data to their cortical sources.
Combining these methods allow scientists to perform more high level analyses, among others:

- individual alpha analysis
- finding peak frequencies over cortical sources
- cross frequency coupling in source space
- cortical oscillator networks
- multivariate pattern analysis on time-frequency transformed cortical data

While MNE-Python provides broad support for Time-Frequency Analysis as well as Source Level Projection, combining these two methods proves difficult. Analysing time-frequency data at source level currently depends on older and more limited functions, covering only parts of the functionality that is available for sensor-level data. For example, with current

functions, only Wavelet Power Analysis and Multitaper PSD Analysis can be performed in source space but full Multitaper Analysis or Stockwell Transform is not doable.
Goal of this GSoC Project will be the integration of source space data containers into the existing sensor-level time-frequency analysis functions, to fully take advantage of both existing structures.

Specific Example:

One example for an neurophysiological analysis involving time-frequency transformed data in source space would be:

Rassi E, Wutz A, Müller-Voggel N, Weisz N. 2018. Pre-stimulus feedback connectivity biases the content of visual experiences [preprint]. bioRxiv doi: 10.1101/437152

Applying spectral and time-frequency analyses on cortical sources, and especially advanced methods like calculating frequency domain Granger analyses between cortical sources can require very specific applications of time-frequency methods, tapering functions, window ranges etc.
Further, giving researchers the possibility to investigate and visualize e.g. different frequency ranges without restrictions can significantly facilitate the research process.
The proposed changes of this project will allow researchers to flexibly apply various types of time-frequency transforms after `mne.SourceEstimate` instance objects were created, and exhaust their respective options to the same extent as they are able when doing sensor space Time-Frequency Analysis. In addition, new SourceTFR objects will enable researchers to flexibly handle and visualize time-frequency data in source space, and will thereby create a solid basis for the further application of more sophisticated analyses on top of it.

Goals:

Enabling full source level Access to TFR functions.

Users should be able to pass `VolVectorSourceEstimates` and `VectorSourceEstimates` into the existing `mne.time_frequency.tfr_xy()` functions and get a time-frequency transformed object in return, while being able to apply the same parameters as for sensor level data. This includes not only the processing of the time-frequency transforms themselves, but also accompanying object information.

Introducing SourceTFR Class(es).

MNE TFR functions currently return `AverageTFR` and `EpochsTFR` objects, which are designed to store sensor level, but not source level TFR data. At least one `AverageSourceTFR` class will be needed, capable of containing single epoch/ averaged time series of time-frequency transformed source level data. Hereby, volume and surface Estimates might be represented by one `AverageSourceTFR` class, passing the information of volume/surface representation further to e.g. plotting functions.
Since `SourceEstimate` objects in MNE are generally represented as single epoch/ averaged time series, the `AverageSourceTFR` would satisfy the user's basic needs for TFR representations of `SourceEstimates`. However, a further step could be to introduce a `EpochsSourceTFR`, which could pose an equivalent source class to `EpochsTFR`.
SourceTFR objects should come with similar methods as `SourceEstimate` objects, while the focus will certainly lie on their visualization.

Visualizing Time-Frequency Transformed source space data.

Plotting of SourceTFR objects will be an important feature. A simple approach would be to let the user specify which frequencies should be plotted, and convert single frequencies into representations that can be plotted by the existing `SourceEstimates` plotting functions `plot_source_estimates` and `plot_volume_source_estimates`.
However, it would be highly desirable to work with pysurfer and create new types of plots, allowing the user to interact with the plotted frequency using a GUI.
For this purpose, the `mne.viz.TimeViewer` could be copied and adapted into a `FrequencyViewer` or a `TimeFrequencyViewer`, which would enable the user to switch between frequencies within the plotting interface.

Handling different kinds of inputs flexibly.

`SourceEstimate` objects (as well as Time-Frequency transformed objects) can come in large data sizes. To handle these sizes, `SourceEstimate` data can be stored using a kernel trick, where the actual data array is recreated from two smaller arrays (`kernel` and `sensor_data`). Multiple `SourceEstimates` (in contrast to `mne.Epochs`) are stored as lists or generator objects.
Applying Time-Frequency Functions on `SourceEstimates` will require handling these types of containers as input for tfr functions, and might as well use the same structures as output to omit huge object sizes.
For this purpose, a straightforward way to start could be:

```
tfr_xy([list_input], …)          returns     [list_output]
tfr_xy(single_obj_input, …)      returns     single_obj_output
tfr_xy(kernel_obj_input, …)      returns     kernel_obj_output
tfr_xy([generator_input], …)     returns     [generator_output]
```

At the same time, the TFR argument to average data should be maintained for lists as well:

```
tfr_xy([list_input],…,              returns      single_obj_output
          average=True)
```

And in a similar way, TFR functions will need to be able create an inter-trial-coherence `AverageSourceTFR`, given lists as an input:

```
tfr_xy([list_input], …,             returns      power_single_obj,
          average=True,                          itc_single_obj
          return_itc=True)
```

Of course, this approach might be enhanced to allow users to pass arguments on whether `[list_inputs]` should be returned as `[list_outputs]`, `[generator_outputs]`, or possibly even as the previously named `EpochsSourceTFR`.

Documenting functions and creating tutorials.

Of course, introducing or changing functions/objects in MNE-Python will require comprehensive documentation. This will be carried out in accordance to MNE-Python's guidelines, using the numpy docstring standard.
Further, a tutorial will help users to understand the undertaken changes, and give them examples on how to use them for their own research. For this purpose, the `mne.datasets.somato` dataset could be used, to show comparisons between the different Time-Frequency Transforms on source projected data. Additionally, examples on how to calculate the inter-trial-coherence from multiple `SourceEstimates` could be implemented.

Possible Project Timeline:

Before Coding Start:
- Investigate and create plans for possible obstacles, especially concerning time-frequency functions, but also plotting
- Fully discuss desired additions to time-frequency data handling and to SourceTFR methods

Before Mid of June:
- Create `AverageSourceTFR` class, including basic attributes and methods (with no or only simple plotting)
- Get TFR functions to allow `SourceEstimates` as input

Before End of June:

- Get Multitapers to return power with TFR Functions
- Get Stockwell to return power with TFR Functions

Before Mid of July:

- Enable Processing Lists/Generators
- Create SourceTFR visualization for Volume and Surface representation, possibly including a `TimeFrequencyViewer` GUI

Before End of July:

- Enable Processing `(kernel, sens_data)` data
- Get Multitapers, Stockwell and Wavelet to return power/itc with TFR Functions

Before End of August/ Final Week:

- `EpochsSourceTFR`?
- Other possible additions
- Documentation
- Tutorials/examples