Enhance statistical inference using linear regression in MNE-Python

Proposal by José Carlos García Alanis
Mentors: D.E., J.S.

Introduction:

In statistics, linear regression is typically used for modelling relationships between one (simple linear regression) or a series (multiple linear regression) of predictor variables and a response variable. In particular, by determining the strength of the relationship between predictors and response variables, linear regression algorithms can explain variation in the response variable, which can be attributed to variation in the predictors, allowing the identification of variables and/or subsets of data that contain relevant information about the response variable.

Implementation in MNE:

To date, linear regression functionality in the premier toolbox for analysing neural time series in Python, MNE, is capable of handling designs mostly characterised by the introduction of categorical predictors, to explain variation in brain activity, based on ordinary least squares estimation.

Basic overview of implementation of linear regression in MNE:

At the moment, the `mne.stats.linear_regression`() function (in the following referred to as `linear_regression` for convenience) in MNE takes an MNE-python data structure (e.g., an array-like object of epoched neural time series data) enriched with metadata about the data to be regressed, such as of number of trials, sensors and time points for regression. In addition, the `linear_regression` function takes a two-dimensional `numpy.array` (number of observations/trials x number of predictors) as `design_matrix` argument. This design matrix and the neural data are then forwarded by the `linear_regression` function to NumPy's `numpy.linalg.lstsq()` function to compute a least square solution of the data given the design matrix. MNE's `linear_regression` function then returns a dictionary of named tuples (one for each predictor/column in the design matrix) containing the regression coefficients and p-values among other results.

Even though this approach can be used to inspect relationships between a wide variety of predictors and brain activity data, the limited options for specifying more complex regression models, such as those based on robust and hierarchical estimation algorithms (see for instance https://osf.io/hdxvb/) are currently preventing users from making use of the functionality of MNE's `linear_regression` at a larger scale, and from using more elaborate regression tools commonly implemented in multiple scientific fields for which MNE is relevant.

In addition, documentation with examples for commonly implemented designs in cognitive neuroscience, as well as auxiliary code that allows further improving and inspecting these designs and their respective results are still missing.

Project goals:

The goal of the GSoC project is to extend the functionality and inference options of the `linear regression` module in MNE-Python, as follows:

1. The project aims at using NumPy and SciPy to expand the `linear regression` functions currently available in MNE. Here, our goal is to provide a set of functions for specifying and testing variants of the linear regression framework, that are commonly used by the neuroscience community. In addition, the project aims at improving the current MNE `linear regression` API to facilitate the interface with these tools and allow, at least to some extent, for a greater compatibility between more complex research questions and MNE's `linear regression` module. For example, one of the project goals is to facilitate the implementation of a robust regression estimation algorithm, which constitutes an extension of the ordinary least squares approximation currently implemented in MNE and can be used to account for outliers or highly influential observations in a dataset. One further goal is to enable the specification of multi-level or hierarchical linear models, which are particularly appropriate when data are organised at more than one level of the experimental design (i.e., single-trial level vs. individuals or groups).
2. Due to the importance of these methods for a broad spectrum of research questions, a growing selection of regression toolkits have been developed and are available as Open software (see for instance scikit-learn in Python https://scikit-learn.org/stable/modules/linear_model.html, MASS https://cran.r-project.org/web/packages/MASS/index.html, and lme4 https://cran.r-project.org/web/packages/lme4/index.html in R to name some). One further goal of the project is to provide and API for interacting with these tools, allowing users for the exploration of more complex designs, such as those fitted to analyse multi-level problems. For this purpose, the project aims at using probabilistic programming tools such as Pymc3 (https://pymc3.readthedocs.io/en/latest/) to provide a set of examples for more complex statistical models, such as those that make use of Bayesian inference for determining relationships between predictors and response variables.
3. In addition, the project aims at providing a set of functions for residual and parametric bootstrapping methods. These tools will allow users for a more general uncertainty estimation and inference in cases when linear models are regularized and standard mathematical procedures for inference are not available.
4. Finally, the project aims at providing a platform for restructuring data sets to fit the format required by common linear regression modules, which will also be validated on open data resources, such as the LIMO (https://datashare.is.ed.ac.uk/handle/10283/2189) and LEMON datasets (http://fcon_1000.projects.nitrc.org/indi/retro/MPI_LEMON.html).

Project Timeline:

During the period of community bonding (May 6 to May 27, 2019) I will focus on making preparations for implementing alternative methods of estimation for linear and non-linear regression functionality and discussion of these with my mentors and other members of the MNE-community. In particular, I will focus on surveying what kind of model specifications are most common in the neuroscience community and what tools (e.g., NumPy, SciPy) can be

used to extend the `linear_regression` function and the `mne.stats` module in MNE to be able to handle these model specifications.

1. **May 27 - June 9 (Week 1 & 2):**
    - Conduct research on possible implementations of the linear regression framework that might be of particular interest for the project (e.g., robust regression)
    - Analyse possible ways, in which we can extend the `linear_regression` function to handle multiple ways of estimation of the linear regression parameters. Here, one option could be the addition of an `estimator` argument to `linear_regression` to activate fall-back functions that take care of the specific estimation steps.
    - *Implementation*:
        - Write generic functions to simulate simple array-like data sets of neural time series data with multiple design elements (e.g., predictor variables) to be used for validation of new regression tools.
        - Write a function to create and visualise a design matrix (see for instance EEG 1st level in https://github.com/LIMO-EEG-Toolbox/limo_eeg/wiki). Here, one option could be that the function accepts metadata from an epochs array in MNE and outputs a design matrix than can be fed to the `linear_regression` module in MNE to use for estimation.
        - Use simulated data sets to create and visualise multiple design matrices.
    - *API*:
        - Write code example on how to use metadata to create a visualise design matrices.
        - Addition of an implementation for `mne.simulation.simulate_epochs()` for testing purposes. Should behave similar to `mne.simulation.simulate_evoked()`, see: https://martinos.org/mne/stable/generated/mne.simulation.simulate_evoked.html.
        - One further possibility would be to add a `mne.stats.make_design()` function that takes a `data` argument, i.e., an array-like object of epoched data (e.g., such as returned by `mne.Epochs()`). In addition, the function should take a further argument `design`. The latter could be specified to be ‘metadata’, indicating the function should use the `mne.Epochs.metadata` to create the design matrix. Alternatively, a two dimensional array-like object (e.g., a `pandas.DataFrame`) can be provided. The function should return a two-dimensional `numpy.array` containing columns for each predictor in the design as well as an intercept.
            - An additional function could be `mne.stats.plot_design()`, which makes use of the

design matrix to returns a visualization of a design matrix as instance of `matplotlib.figure.Figure`.

2. **June 10 - June 23 (Week 3 & 4):**
    - Test what measures could be best fitted to extend the functionality of `linear_regression` in MNE. For example, robust regression would need a robust estimator and control for outliers in dataset. One option could be to use the interquartile range of the data distribution (for instance using `scipy.stats.iqr()`; https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.iqr.html).
    - *Implementation:*
        - Preparation of code to allow a standardised input way into the linear regression module; e.g.:
            - Extract data from the given structure and put it the right format (i.e., long format; for instance, using `numpy.reshape()`,see https://docs.scipy.org/doc/numpy/reference/generated/numpy.reshape.html or `pandas.wide_to_long()`, see https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.wide_to_long.html) to allow compatibility with multiple estimation methods.
            - Get design matrix for experimental design (e.g., by calling a previously created `make_design()` function; see above).
            - Check which estimation is desired and activate fall-back functions to carry out estimation.
            - Return a standardised `linear_regression` object that contains the estimation results (e.g., betas) and can be used for further analysis (e.g., prediction of data).
    - *API and Documentation*:
        - Addition of `make_design()` to MNE's `linear_regression` function.
        - Addition of function that checks if data is in the right format (i.e., long format) and rectifies the format if needed.
        - Further possibility: Addition of `estimator` argument to `linear_regression` function to select the type of estimation algorithm to be used.
        - Documentation of the progress as example code to test the additions on simulated data (e.g, implementation of newly created `mne.simulation.simulate_epochs()` in documented example).
3. **June 24 - July 21 (Week 5 - 8):**
    - Write code and fall-back functions for specific the estimation methods established in the previous weeks.
    - *Implementation:*
        - For robust regression, e.g.:
            1. Write code to find M-estimator, which will be used to lower the influence of outliers in the data set.
            2. Adjust regression residuals.
            3. Carry out matrix multiplication.

        - Test functions.
        - Debug.
        - Start writing function for bootstrap of the regression results.
        - Start formulation of a set of functions that allow interaction with other packages (e.g., Pymc3) for more complex models.
    - *API and documentation:*
        - Provide weekly updates on progress.
        - Documentation.
        - Start integration of linear regression API.
            - Extend OLS-approximation in `linear_regression` function to accept further estimation methods (e.g., add `estimator` argument). Here, the backend functions will be used for estimation.
        - Start integration of boostrap functionality for `linear_regression` to MNE API.
            - E.g., add `bootstrap` argument for `linear_regression` function.
        - Test and debug.
        - Submission of code for ***first evaluation (June 24 - 28, 2019)***.
4. **July 22 - August 4 (Week 9 & 10):**
    - *Engineering:*
        - Unit testing of the newly created functions.
        - Validation on functions on open datasets.
    - *API and documentation:*
        - Improve and wrap-up documentation.
        - Continue testing on open data sets and bring code to examples for users.
        - Submission of code for ***second evaluation (July 22 - 26, 2019)***
5. **August 5 - August 18 (Week 11 & 12):**
    - Final evaluation and submission of code to google *for* ***final evaluation (August 19 - 26, 2019)***.
    - *Implementation & Engineering:*
        - Code clean up, fix bugs add comments and, if possible, fine tuning to improve functionality.
        - Unit testing.
        - Validation of the functions and testing on open datasets (e.g., the LIMO dataset).
    - *Interface & Documentation:*
        - Check integration in API, Improve and review documentation.
        - Write examples with auxiliary code explaining implementation of the added functionality of `linear_regression`.

Blog for this project:

I will be writing blog post to document my progress on every stage of the GSoC timeline. In the blog, I will be sharing my ideas and suggestions made by my mentors. In the blog, I will write about difficulties and challenges I have encountered along the way, while discussing

different strategies I tried to achieve a solution as well as advantages and disadvantages of differents ways of implementation.

I didn't add a link to that blog to this proposal, because I don't have much experience with technical blogging, so prior to creating one I would like to consult the MNE-community and my mentors on which blogging platform to choose.

Why do I want to work on this project?

The projects aims to build tools that I myself would like to implement in my research. I am currently working towards a PhD in the field of psychophysiological research and am interested in finding ways to describe individual variation in brain signals in different situational settings.

For me, this also a great opportunity to collaborate with the MNE-community and expand my set of skills in the programming of Python and analysis of neural data. One thing I'm excited about is deepening my understanding of the actual steps necessary to specify and formulate good and useful models for data analysis.

Benefits to the community:

Many studies aim at finding relationships between environmental, personal features and patterns of brain activity. Without the ability to customise designs for statistical testing data analysis can lead to suboptimal conclusions. Extending MNE's capabilities in the field of linear regression will allow more robust and versatile pipelines for data analysis that make adequate assumptions to deal with different types of models.

In the past, many users have expressed interest for wider implementation capabilities of MNE in the field of linear regression. Thus, the project might motivate users to analyse their data with MNE and contribute by specifying designs for linear regression themselves.

About me:

https://github.com/JoseAlanis/cvlatex/blob/master/CV.pdf

Contributions to MNE:

https://github.com/mne-tools/mne-python/pulls?q=is%3Apr+author%3AJoseAlanis+is%3Aclosed