[0] emcee: Be the life of the party

Rohan Panchwagh – 5/15/2025

[1] The emcee package is a package primarily written in Python. Its main goal is to generate the posterior distribution of parameters for a model given an input likelihood function and data. This is primarily used when you have extremely noisy data, non-Gaussian/non-analytic error, or when you don't trust your uncertainty estimates. It is also better at handling high-dimensional problems. In comparison with something like maximum-likelihood estimation or least-squares, emcee allows you to do error propagation on parameters even if you distrust the error on the individual data points (for example, if they are underestimated, then doing Gaussian error propagation would give an underestimated uncertainty).

[2] I selected this package because I use Monte Carlo in my research and wanted to learn more about Markov Chain methods (in particular, how they are used to sample distributions).

[3] The emcee package was first released in 2012, by Foreman-Mackey, Hogg, Lang, and Goodman, based on the affine-invariant ensemble sampler proposed by Goodman and Weare in 2010. Before emcee, the main package used was PyMC (though it wasn't as scalable as emcee). Current packages that do something similar are NumPyro (compatible with GPU acceleration), and pymcmcstat (adapted from a MATLAB toolbox). The current emcee version is 3.1.6.

[4] emcee is still maintained by the original authors, primarily Foreman-Mackey. The CONTRIBUTING.md file in the emcee GitHub contains instructions on how to contribute.

[5,6] The emcee package is extremely easy to install. Simply by using the command "!pip install emcee", the package can be installed and is ready to go. However, installing visualizing tools (such as corner) can make understanding the marginal posterior distributions easier, and is highly recommended.

[7] The source code can be found on the GitHub, in the "src/emcee" file. The contained files are mainly python files, with a test file, moves file, and a backend file.

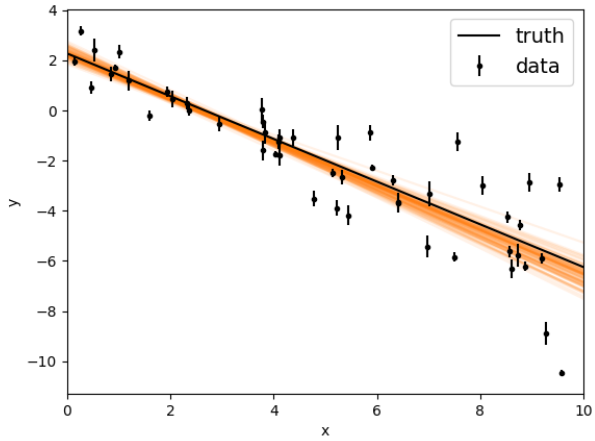
[8] The radvel (radial velocity fitter for exoplanets) and bilby (gravitational waves) libraries use (or are compatible with) the use of emcee.

[9] The code is pure python, meaning that it can be run as a python script in any development tool, as well as in a Jupyter notebook (which is where you would run it to get visualizations).

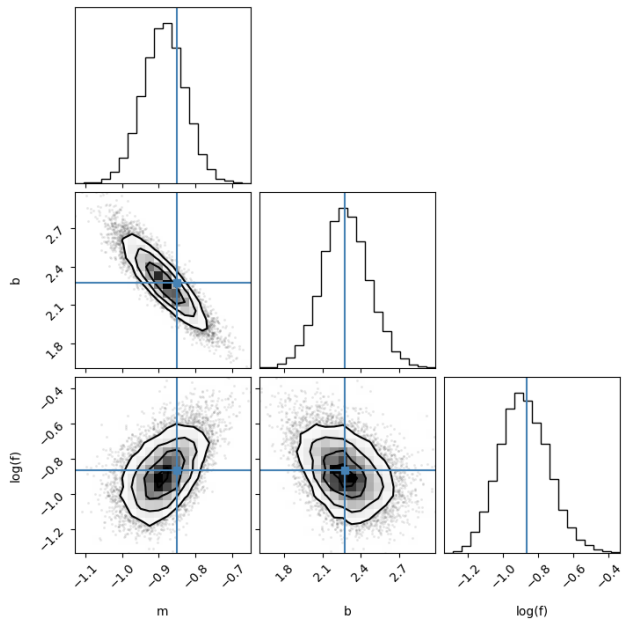
[10] The code is shown in the example Jupyter notebook.

[11] emcee does not contain its own plotting software; typically, matplotlib or corner are used for visualization (depending on what one needs to visualize).

[12] Here is a figure showing samples from the output parameter distribution, against the true model and the data. To be more precise, samples are taken from the posterior distribution, inserted into the model, and that line is plotted (in orange). Since the orange values are clustered around the true model, the emcee analysis seems to be valid.



The second figure shows the corner plots for our parameters. The diagonal plots give the marginal posterior probability distribution for each parameter individually; the width of this distribution gives the uncertainty in the parameter. The off-diagonal plots show the marginal joint probability density function for a pair of parameters. The shape (elongated vs circular) tells you how the parameters are correlated (for example, m and b seem to be negatively correlated since the shape of their joint pdf is an ellipse slanted downwards). The strength of the correlation indicates a higher magnitude of covariance, while the sign of the correlation gives the sign of the covariance. Each contour represents a confidence interval for the true parameter values. The blue line(s) represent the true value(s).



[13] The emcee package is purely Python, with no dependencies on C/C++/Fortran.

[14] There are two inputs to the package; one is the likelihood function for the data (or typically, the log of the likelihood function for the data), and the other is the data itself. Test data can be generated by drawing from a distribution or using random number generators; or one can use actual data. The notebook data is generated by using a random number generator on a linear model.

[15] The output is a *posterior distribution* of parameter values (more specifically, a set of *samples* from this distribution); instead of returning the "best-fit" value (as done in curve fits), it returns a set of samples from around the maximum-likelihood value, which can be used to estimate the uncertainty in each parameter.

[16] The code provides ways of visualizing the progression of the walkers, as well as calculating the autocorrelation times, to ensure that the entire parameter space is being sufficiently explored (in other words, preventing the walkers from getting caught in local minima). One can essentially "benchmark" by inserting in a known distribution, injecting it with some error, and seeing if the returned posterior distribution matches the true model.

[17] Once you look at the visualization of the walkers "burning in" to explore the parameter space, you can be sure that the posterior distribution will be fairly accurate.

[18] emcee mainly depends on 4 packages: numpy, matplotlib, scipy, and corner. I figured this out by looking at their example code: numpy is used for random number generation and matrix manipulation, matplotlib is used for plotting distributions and model data, scipy

is used to obtain the maximum likelihood fit, and corner is used to visualize the shapes of the uncertainty distributions for each parameter.

[19] emcee has a website with worked examples, as well as comprehensive explanations for each function unique to emcee. There is also more documentation on the GitHub for emcee. Overall, I felt that the website did a good job explaining what each function did, and how each visualization should be interpreted, and the example given was strong enough to enable a user to immediately apply emcee to sample data. However, there wasn't much information on the actual theory behind MCMC, and that needed to be found elsewhere (such as Wikipedia).

[20, 21]

References:

- emcee preferred citation: https://ui.adsabs.harvard.edu/abs/2013PASP..125..306F/abstract
- Goodman and Weare paper: Jonathan Goodman. Jonathan Weare. "Ensemble samplers with affine invariance." Commun. Appl. Math. Comput. Sci. 5 (1) 65 - 80, 2010. https://doi.org/10.2140/camcos.2010.5.65
- corner: https://ascl.net/corner
- PyMC: https://ascl.net/pymc
- radvel: https://ascl.net/radvel
- bilby: https://ascl.net/bilby
- NumPyro: https://doi.org/10.48550/arXiv.1912.11554
- pyMCMCstat: Miles, (2019). pymcmcstat: A Python Package for Bayesian Inference Using Delayed Rejection Adaptive Metropolis. Journal of Open Source Software, 4(38), 1417, https://doi.org/10.21105/joss.01417

[22] Using ADS, there are 10,640 citations for the emcee paper. Two of the most recent ones are given here: https://ui.adsabs.harvard.edu/abs/2025NewA..11902390P/abstract and https://ui.adsabs.harvard.edu/abs/2025NewA..11802374N/abstract .

[23] I didn't really have to learn new python methods since almost everything was done using random number generators (numpy) and matplotlib visualizations. The only new thing I learned was corner, but since I was only using it to generate and interpret a plot, I didn't have to have full understanding of exactly how corner works, just how to read it.

[24] I had no exposure to emcee before this project; this is my first time using it. I worked alone to complete this project.