Asa Rishel
May 18, 2025

Final Project Report: LMfit

[1] LMfit: al-**L**ow **M**e to **FIT** this title

[2] The LMfit package aims to improve upon the optimization and fitting capabilities of SciPy. It does so by allowing the user to make use of a new parameter object, which functions similarly to a dictionary in Python. A parameter can have minimum and maximum values, satisfy relationships with other parameters (e.g. A + B + C = 4), and be held constant without needing to change the underlying function being fit. The package also makes it easy to turn any function into a model just by casting it as such with *Model(function)*. And compared to SciPy, LMfit calculates statistics like confidence intervals and the covariance matrix for the fitted parameters more accurately.

[3] I selected this package because I am going to use it extensively in my work this summer. I am working in a math research lab at UMD with Dr. Abba Gumel that applies mathematical techniques to problems in epidemiology and biology more broadly. Currently, I am researching how to model the way a disease outbreak affects the mental health of the population using delay differential equations. In principle, this can be used to model the interaction between any infectious and non-infectious disease, but I am looking specifically at how COVID-19 affected depression. My work up to now has been to design the differential equations which govern movement between different disease states (an example path might be susceptible → mild infection → mild infection and depressed → recovered and depressed), and it is a model with roughly twenty parameters. I have found values for common parameters (e.g. COVID-19 infectivity), but now I need to fit values for the more niche parameters in my model (e.g. the rate at which people with long-COVID develop depression), and lmfit was specifically recommended to me as a package that would be useful to streamline this process. So this project functions as somewhat of an introduction to a tool I will be using soon in my research.

[4] The LMfit package was created in 2009 by Eric Lebigot. LMfit was designed to extend the capabilities of SciPy by making it feasible to fit large amounts of parameters, giving more control over parameter selection, and providing a function to calculate confidence intervals for values on these parameters. I installed version 1.3.2 of LMfit.

[5] As of 2024 ownership of LMfit has been in the hands of the LMfit GitHub Organization. It is comprised of Dan Allan and Matt Newville, neither of whom are the original author Eric Lebigot.

[6, 7] LMfit is simply installed through pip in a Jupyter notebook by running *!pip install lmfit* in a cell.

[8] The source code can be found in the GitHub repository *lmfit-py*.

[9] LMfit is not a dependency for many other python packages, but it is often combined with emcee to determine uncertainties on parameter values. A function can be fit with LMfit and then have the parameters given to emcee to run a Markov chain Monte Carlo simulation on the parameter space and better select the appropriate values.

[10] The code can be run in a Jupyter notebook.

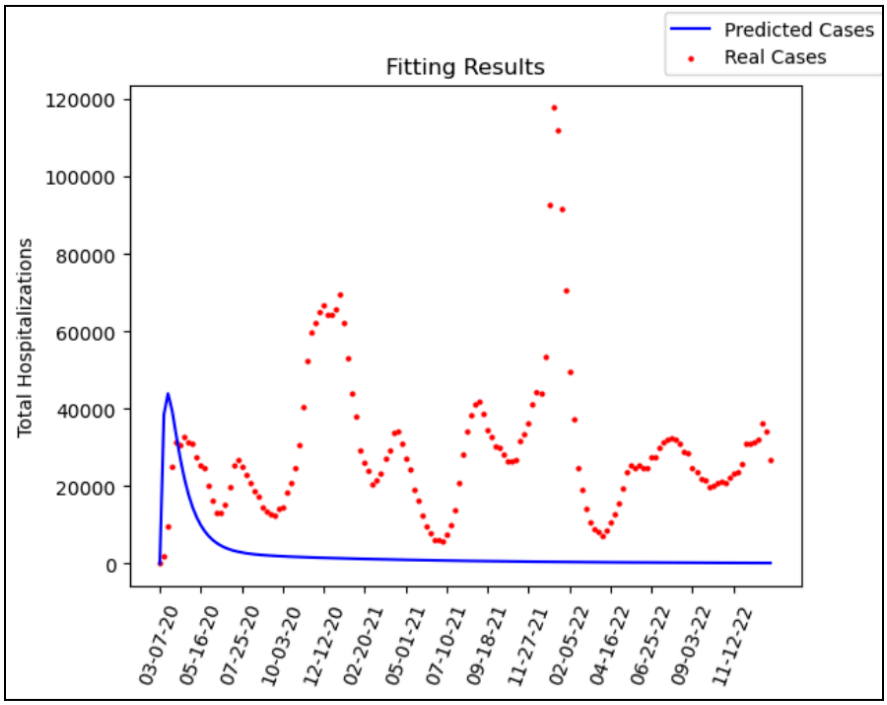
[11] The modeling and fitting functions of LMfit can be performed like so:

Fitting with LMfit

```
COVID_model = lmfit.Model(spread_results, independent_vars = ['t'])

result = COVID_model.fit(hospitalizations,
                         params = params,
                         t = times,
                         max_nfev = 1000,
                         method = 'leastsq')

prediction = result.best_fit

print(result.fit_report())
```

[12, 13] The package can be used to create figures, but it is not necessarily built in. By calling *ModelResult.best_fit*, the best fit parameter values can be evaluated to give predictions. The following figure shows the limitations, however, of the model for more complicated datasets, as

in the long run, the performance of the fit here falls off significantly:



[14, 19] LMfit is a pure python package built off SciPy and Numpy. Some functions like *ModelResult.plot_fit* or *ModelResult.plot_residuals* also use MatPlotLib. All of these dependencies are stated on the GitHub page.

[15] LMfit requires as its input a function and a data set. This function has certain parameters which the user can choose to be fixed or fitted. The data set is the data to which the input function is being fit and it must be supplied by the user. There are also optional keyword arguments the user can supply such as the method used for fitting (e.g. least squares) or the maximum number of function evaluations allowed.

[16] LMfit outputs a *ModelResult* object which consists of the best-fit parameters it finds (per user specifications like parameter maximums/minimums), the statistics and uncertainties of the fit, residuals, etc. All of these can be accessed through a simple call; for example, *ModelResult.params* will return the best fit parameters.

[17, 18] One of the key aims of LMfit was to provide more statistical information to provide the user with insight into how parameters were selected. Uncertainties on specific parameter values can be called with *ModelResult.uvars* or they can be calculated separately from the covariance matrix which can be accessed by *ModelResult.covar*. This allows the user more control over the selection of parameters; the built-in uncertainty calculations make it easy to

find a range of acceptable parameter values. Benchmarking is not built in but it can easily be done manually by comparing how long a result takes or how accurate the fit is with another package like SciPy.

[20] The package has extensive documentation on GitHub that makes it easy to learn.

[21] The preferred citation method is via the DOI 10.5281/zenodo.12785036. This is specified in *lmfit-py/CITATION.cff* in the GitHub repository.

[22] References:

1. lmfit/uncertainties: Transparent calculations with uncertainties on the quantities involved (aka "error propagation"); calculation of derivatives.
2. Non-Linear Least-Squares Minimization and Curve-Fitting for Python — Non-Linear Least-Squares Minimization and Curve-Fitting for Python

[23] Using ADS, I found seven papers that cite LMfit via its DOI. It is cited in:

1. A paper on the new discovery of a massive Quiescent Galaxy https://ui.adsabs.harvard.edu/abs/2025ApJ...983...11W/abstract
2. A physics toolbox for laser spectroscopy, the Python package qspec https://ui.adsabs.harvard.edu/abs/2025CoPhC.31109550M/abstract

[24] The class was sufficient to get through this project. LMfit is similar enough to SciPy that I felt familiar with how to use the tool after a quick read through the documentation.

[25] I have no prior experience with LMfit specifically, although I have worked with differential equations similar to the ones in this project.