The Journal of Open Source Software

RAFF.jl: Robust Algebraic Fitting Function in Julia

Emerson V. Castelani[1], **Ronaldo Lopes**[1], **Wesley V. I. Shirabayashi**[1], **and Francisco N. C. Sobral**[1]

1 Department of Mathematics, State University of Maringá, Paraná, Brazil

DOI: 10.21105/joss.01385

Software

- Review
- Repository
- Archive

Submitted: 12 March 2019
Published: 02 July 2019

Summary

`RAFF.jl` is a Julia package for the adjustment of a function to a dataset coming from some experiment. This package is an alternative to classical adjustment techniques such as linear and nonlinear regression. The goal of this package is to find robust adjustments free from the influence of possible outliers (discrepant points of the adjustment).

Motivation

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function whose mathematical description is not available. This function can be, for example, a black-box, a proprietary computer program or an experiment. Suppose that a dataset $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ is available, where y_i is an approximation of $f(x_i)$ (from an experimental procedure, numerical approximation, etc.) and we want to approximate f by a known model ϕ. Model ϕ can be defined as $\phi(x, \theta)$, where x are the n independent variables of f and θ represents some parameters of ϕ. `RAFF.jl` (Robust Algebraic Fitting Function) is a Julia package developed to find parameters θ for ϕ in order to adjust it to the observed values S of the unknown function f. Following Liu & Wang (2008) and Keleş (2018), in general, the adjustment can be related to

1. Classical least squares (algebraic fit): which considers the sum of deviations of type $|\phi(x_i, \theta) - y_i|^2$, also known as regression;
2. Orthogonal least squares (geometric fit): which considers the sum of deviations of type $\min_x \|(x, \phi(x, \theta)) - (x_i, y_i)\|^2$ (orthogonal projection on the curve to be adjusted).

`RAFF.jl` was developed to solve a generalization of the first case.

Linear and nonlinear regression is essentially the adjustment of mathematical functions to data and is a problem that appears in many areas of science. When data comes from real experiments, non-expected errors may cause the appearance of outliers, which might be responsible for causing the regression calculated by sum of deviations to result in misleading approximations. Regression is strongly connected to Statistics but practical methods to detect outliers are not very common. Motulsky & Brown (2006), for example, develop a method for outlier detection based on the assumption that the error follows a Lorentzian distribution around the function and use nonlinear regression based on least squares. `RAFF.jl` provides automatic detection of outliers using a voting system. It is an optimization-based package, based on algorithms for Lower Order-Value Optimization (LOVO) which were introduced by Andreani, Dunder, & Martínez (2005) and revisited by Andreani, Martínez, Martínez, & Yano (2009). Recently, Martínez (2012) performed a complete review about LOVO problems considering theoretical aspects of algorithms to solve it and potential applications.

Background

To elucidate the essence of how `RAFF.jl` works, let us detail some aspects related to the LOVO problem and its resolution. Let us consider m functions $F_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., m$. Given $\theta \in \mathbb{R}^n$, we can sort the set $\{F_i(\theta), i = 1, ..., m\}$ in ascending order:

$$F_{i_1(\theta)}(\theta) \leq F_{i_2(\theta)}(\theta) \leq ... \leq F_{i_m(\theta)}(\theta).$$

Considering a value $1 \leq p \leq m$, we can define the LOVO function as

$$S_p(\theta) = \sum_{k=1}^{p} F_{i_k(\theta)}(\theta)$$

and the LOVO problem as

$$\min_{\theta \in \mathbb{R}^n} S_p(\theta).$$

Assuming that $F_i, i = 1, ..., m$ are continuous functions we have that S_p is a continuous function, but assuming that F_i's are differentiable functions we cannot conclude that S_p is differentiable. This can be seen by reformulating the LOVO problem as follows. Denoting $\mathcal{C} = \{\mathcal{C}_1, ..., \mathcal{C}_r\}$ as the set of all combinations of $\{1, ..., m\}$ taken p at time, we can define for each $i \in \{1, ..., r\}$ the following function

$$f_i(\theta) = \sum_{k \in \mathcal{C}_i} F_k(\theta)$$

and

$$f_{min}(\theta) = \min\{f_1(\theta), ..., f_r(\theta)\}.$$

It can be observed that, for a given θ, $f_{min}(\theta)$ is obtained by a combination $\mathcal{C}_j$ which contains the smallest sum of p elements of the set $\{F_i(\theta), i = 1, ..., m\}$. Therefore $f_{\min}(\theta) = S_p(\theta)$ and, consequently, the LOVO function is non differentiable. The LOVO problem description can become more clear by considering an example. In this sense, let us consider the dataset given by

x	y
-0.5	0.119447
0.0	0.3
0.5	0.203551
0.75	0.423998

and the model defined by $\phi(x, \theta) = \theta(\sin(x) + \cos(x))$. Naturally, we have $m = 4$ and let us consider $p = 3$. The F_i's can assume diferent forms. To leave the example closest to our approach, let's consider F_i's as:

$$\begin{aligned}
F_1(\theta) &= (0.119447 - \phi(-0.5, \theta))^2, \\
F_2(\theta) &= (0.3 - \phi(0.0, \theta))^2, \\
F_3(\theta) &= (0.203551 - \phi(0.5, \theta))^2, \\
F_4(\theta) &= (0.423998 - \phi(-0.75, \theta))^2.
\end{aligned}$$

Since $m = 4$ and $p = 3$, we have 4 possible subsets with 3 elements each from set $\{1, 2, 3, 4\}$:

$$\mathcal{C}_1 = \{1, 2, 3\}, \mathcal{C}_2 = \{1, 2, 4\}, \mathcal{C}_3 = \{1, 3, 4\} \text{ and } \mathcal{C}_4 = \{2, 3, 4\}.$$

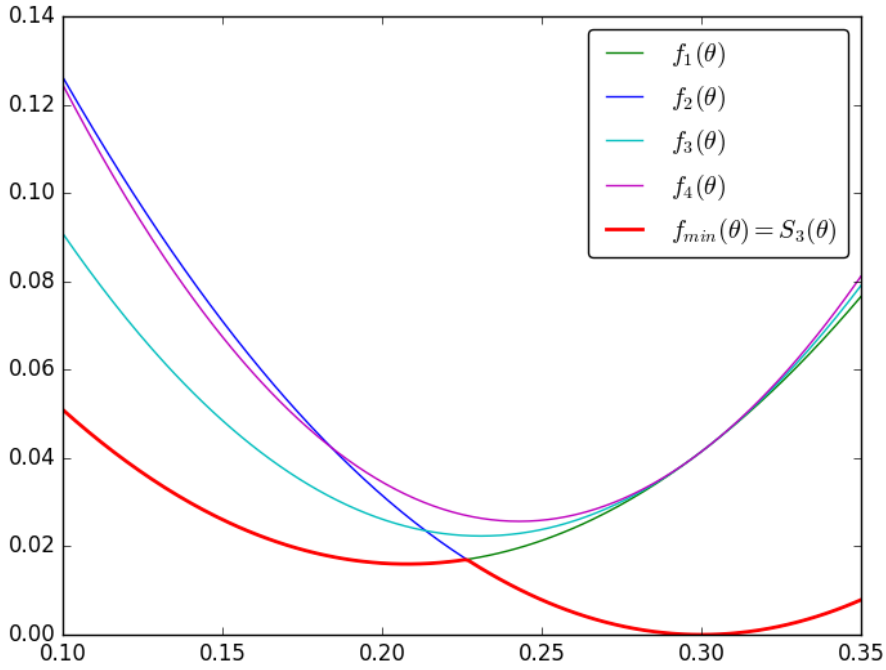

Figure 1: The red function represents the LOVO function. Observing the interval $[0.2, 0.25]$ we can note a singular point even considering f_1, f_2, f_3 and f_4 as differentiable functions.

Thus, associated to each $\mathcal{C}_i, i = 1, ..., 4$, we can define function f_i as follows

$$\begin{aligned} f_1(\theta) &= F_1(\theta) + F_2(\theta) + F_3(\theta), \\ f_2(\theta) &= F_1(\theta) + F_2(\theta) + F_4(\theta), \\ f_3(\theta) &= F_1(\theta) + F_3(\theta) + F_4(\theta), \\ f_4(\theta) &= F_2(\theta) + F_3(\theta) + F_4(\theta), \end{aligned}$$

and consequently,

$$f_{min}(\theta) = \min\{f_1(\theta), f_2(\theta), f_3(\theta), f_4(\theta)\} = S_3(\theta).$$

As previously pointed out, this function is continuous but it is not differentiable as illustrated in Figure 2.

Andreani et al. (2009) introduced line search methods and handled the possible singularities in a clever way, using the following approximation for $\nabla f_{min}(\theta)$

$$\nabla f_{min}(\theta) = \nabla f_i(\theta),$$

where $i \in \mathcal{I}(\theta) = \{k \in \{1, ..., r\}; f_k(\theta) = f_{min}(\theta)\}$. This approach can naturally be extended for second order derivatives.

An important point for practical purposes is when we consider the LOVO problem with $p = m$ and $F_i(\theta) = (\phi(x_i, \theta) - y_i)^2$. In this case, the LOVO problem coincides with classical least squares and, consequently, it can be seen as a generalization of the least squares problem. When $p < m$ and $F_i(\theta) = (\phi(x_i, \theta) - y_i)^2$, the solution θ provides a model $\phi(x, \theta)$ free from influence of the $m - p$ points with the highest deviation. The number p can be interpreted as the number of trusted points, that is, $m - p$ possible outliers were identified.

One of the most usual ways to solve the problem of nonlinear least squares is by using the Levenberg-Marquardt method (Moré, 1978). This method is a first-order method, where derivatives of the model ϕ *with respect to* θ are used to compute the gradient of the objective function in the associated least squares problem. The reason for the wide use of Levenberg-Marquardt method is, in general, associated with quadratic convergence properties even using only first-order derivatives. In this direction, it is relevant to ask about Levenberg-Marquardt-based methods to solve LOVO problems in the context of adjustment functions.

`RAFF.jl` implements a Levenberg-Marquardt algorithm in the context of LOVO problems, i.e., it solves the problem of minimizing $f_{min}(\theta)$, where $F_i(\theta) = (\phi(x_i, \theta) - y_i)^2$, for $i = 1, \ldots, m$. In this sense, first-order derivatives are necessary and the same strategy of Andreani et al. (2009) is used. It uses first-order derivatives of the model ϕ with respect to θ to approximate the gradient of $f_{min}(\theta)$, which is a non differentiable function. Moreover, LOVO problems have the limitation that the number p of possible trusted points needs to be given by the user. `RAFF.jl` solves this limitation by implementing a voting system. In this voting system, several LOVO subproblems are solved with different values for p, the number of possible trusted points. Each solution of a LOVO subproblem is associated to a vector parameter θ. The vector parameters are compared against each other using the Euclidean distance, where small distances (using a threshold) are considered the same solution. The parameter θ^* which most occurs among them is declared as the solution.

Functionality

`RAFF.jl` main methods expect as input a dataset of the observed data and a model function, whose parameters one intends to adjust. The model function is a regular Julia function with 2 arguments: θ represents the parameters of the model and x represents the arguments of function f. The following function is an example of a model representing the logistic function

$$\phi(x, \theta) = \theta_1 + \frac{\theta_2}{1.0 + \exp(-\theta_3 x + \theta_4)}.$$

The observed data can be represented by the following table:

x	y
0.0000	1166.0892
3.3333	1384.4495
6.6666	4054.1959
10.0000	2692.4928
13.3333	3011.5096
16.6666	3882.4381
20.0000	4612.4603
23.3333	6605.6544
26.6666	5880.1774
30.0000	5506.3050

In this example, the true function was given by

$$f(x) = 1000 + \frac{5000}{1.0 + \exp(-0.2x + 3)}.$$

The observed data was generated as random normal perturbations around the graphic of f and is shown in Figure 1. The dots and triangles represent the observed data, where the red triangles were manually set to be the outliers. Using the least squares technique with the

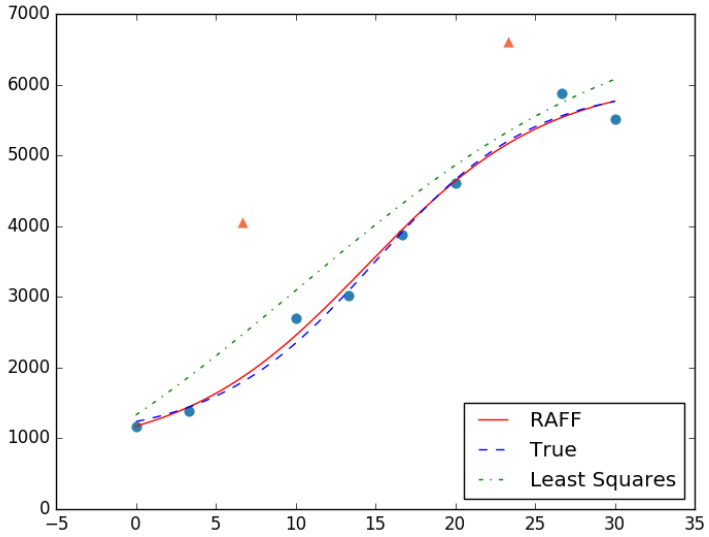

Figure 2: Points representing the logistic function. The red triangles are two outliers that should be ignored. The blue dashed function is the true one, while the green was obtained by traditional least squares techniques and the red one was obtained by `RAFF.jl`.

model above, the green function is found. When `RAFF.jl` is applied to the same problem, it correctly identifies the two outliers. The resulting function is depicted as the red one, very close to f.

Additional features

The user may also provide more information to `RAFF.jl`, such as an rough approximation to the expected number of *trusted* observations. Additional methods and options are also available to more advanced users, such as generation of random test data and multistart strategies. First-order derivatives of the model ϕ with respect to θ can also be provided, which results in a faster executing time. When they are not provided by the user, `RAFF.jl` uses Julia's `ForwardDiff.jl` package (Revels, Lubin, & Papamarkou, 2016).

`RAFF.jl` can be run in serial, parallel and distributed environments. Parallel and distributed methods use the native `Distributed.jl` package. The distributed version is a primary-worker implementation that does not use shared arrays, therefore, can be run both locally or on a cluster of computers.

This package is intended to be used by any experimental researcher with a little knowledge about mathematical modeling and fitting functions.

Installation and usage

`RAFF.jl` is an open-source software that can be downloaded from Github. It is a registered package and can be directly installed from Julia's package repository. The whole description for first time usage or its API is available at its documentation.

References

Andreani, R., Dunder, C., & Martínez, J. M. (2005). Nonlinear-Programming Reformulation of the Order-Value Optimization Problem. *Mathematical Methods of Operations Research*, *61*(3), 365–384. doi:10.1007/s001860400410

Andreani, R., Martínez, J. M., Martínez, L., & Yano, F. S. (2009). Low Order-Value Optimization and Applications. *Journal of Global Optimization*, *43*(1), 1–22. doi:10.1007/s10898-008-9280-3

Keleş, T. (2018). Comparison of Classical Least Squares and Orthogonal Regression in Measurement Error Models. *International Online Journal of Educational Sciences*, *10*(3), 200–214. doi:10.15345/iojes.2018.03.013

Liu, Y., & Wang, W. (2008). A Revisit to Least Squares Orthogonal Distance Fitting of Parametric Curves and Surfaces. In F. Chen & B. Jüttler (Eds.), *Advances in geometric modeling and processing* (pp. 384–397). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-540-79246-8_29

Martínez, J. M. (2012). Generalized order-value optimization. *Top*, *20*(1), 75–98. doi:10.1007/s11750-010-0169-1

Moré, J. (1978). The Levenberg-Marquardt Algorithm: Implementation and Theory. *Numerical Analysis*, 105–116. doi:10.1007/BFb0067700

Motulsky, H. J., & Brown, E. R. (2006). Detecting Outliers When Fitting Data with Nonlinear Regression – a New Method Based on Robust Nonlinear Regression and the False Discovery Rate. *BMC Bioinformatics*, *7*(123). doi:10.1186/1471-2105-7-123

Revels, J., Lubin, M., & Papamarkou, T. (2016). Forward-mode automatic differentiation in Julia. arXiv:1607.07892 [cs.MS].