Overall response to reviewer comments.

To the editors:

We thank both reviewers for their helpful suggestions to improve the JuPOETs manuscript. However, in both cases we're very concerned there was some confusion as to the category of this paper. This manuscript was submitted as a software note that described the JuPOETs approach and two distinct example applications of JuPOETs. It was not intended to be a full research paper that exhaustively analyzed each application, or deeply explored the algorithmic aspects of JuPOETs versus other approaches. Reviewer #2 comments clearly demonstrated this confusion "… Yet, I do not think the new software version adds enough technical improvements to deserve a full paper. I would see it as a short note.".

Again, this manuscript was submitted as a software note. We believe it was entered as such in the manuscript submission system. Thus, we feel the reviewers' technical comments were unwarranted given the classification of the manuscript. Nevertheless, we've addressed the reviewer comments below. Our response is indented below each comment (and shown in red). In addition, new references added to the manuscript have been listed. Lastly, new text added to the manuscript (or text that has been modified) is highlighted in the manuscript where applicable.

Toward this confusion, we have globally replaced the phrase "… In this study, …" with "… In this software note,…" to better orient the readers (and reviewers) that this manuscript is not a full length research paper, but instead is a software note.

Reviewer #1:

The paper presents an open source implementation of the Pareto optimal ensemble technique in the Julia programming language. This software tool improves a previous implementation in the Octave language, exploiting the advantages of Julia programming. The tool implements a multi-objective based technique to estimate parameter or models ensembles for robust predictions. I find this tool useful for the community, however, there are some major issues that need to be addressed before publication.

Major comments:
————————

1. (BACKGROUND) The use of a multiobjective approach needs to be justified (and contextualized).

1.1. Justification:

Single objective optimization is broadly (and successfully) used to estimate parameters / identify biochemical models with hundreds/thousands of states and parameters, with training data from diverse sources. Taking into account that multiobjective problems are more difficult to solve than single objective problems, why to use multiobjective approaches in the first place?.

From the paragraph:

"Identification of biochemical models with hundreds or even thousands of states

and parameters may not be tractable as a single objective optimization problem. Further, large models require significant training data perhaps taken from diverse sources, for example different laboratories or cell-lines. These data are often heterogenous, and contain intrinsic conflicts that complicate parameter estimation. Parameter ensembles which optimally balance tradeoffs between submodels and conflicts in training data can lead to robust model performance."

it is not clear what is the advantage of multiobjective approaches (instead of weighting the different data sets). Please justify your choice (multiobjective optimization), including a better explanation of what is meant by "intrinsic conflicts that complicate parameter estimation", and "optimally balance tradeoffs between submodels and conflicts in training data". The aim is to help the reader to decide when it would be convenient to use a multiobjective optimization approach instead of a single optimization one.

We thank the reviewer for this helpful suggestion which will strengthen the case for multiobjective optimization. The central advantage of multiobjective optimization comes from a significant real-world problem when estimating model parameters from noisy data; the need to balance between competing model training objectives. For example, it is common to train signal transduction models on data sets from multiple laboratories, multiple cell lines, or different data sets from the same cell line on different days that have been corrupted by operator or other uncontrollable experimental artifacts. In these cases, multiobjective optimization is a convenient approach, which optimally balances (or tradeoffs) between competing data sets without the need to generate ad hoc objective weighting schemes. Thus, multiobjective optimization can be easier in these common cases than the equivalent single objective problem.

We have updated the second paragraph in the introduction with a new referenced that outlines five reasons for using multi-objective optimization in the identification of biochemical models. Additionally, we have added text/references to the second paragraph suggested by the reviewer regarding the use of multiobjective optimization in synthetic biology.

We have added the following reference to the introduction to further motivate the use of multiobjective optimization:

Handl J, Kell D.B and J. Knowles (2007) Multiobjective Optimization in Bioinformatics and Computational Biology. IEE Trans Comp Biol Bioinform 4:279 - 292.

1.2 Contextualization

Multicriteria (Pareto) optimality concepts are being increasingly used in the context of systems and synthetic biology. The introduction in my opinion is missing some initial discussion and recent examples on successful use of multi-criteria optimization methods in biology:

http://bmcsystbiol.biomedcentral.com/articles/10.1186/s12918-014-0113-3
http://www.nature.com/articles/srep15147
http://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-015-0706-x

We thank the reviewer for suggesting these supporting references, especially with respect to synthetic biology. They have been incorporated into the background section of the manuscript.

2. (IMPLEMENTATION) The problem formulation needs to be rewritten to improve readability. It is hard to follow even for a reader familiar with multiobjective optimization:

-Please first introduce the model equations, then the objective functions and finally formulate the problem (1).
-Please specify the type of model equations, type of decision variables (ODEs, real variables?)
-Please define the trade-off surface
-Please elaborate more on the modified simulated annealing approach you use (please see comment 3)
-Please provide definitions for all the technical terms you use (Pareto optimality, computational annealing temperature,...)
-Please describe the algorithm inputs and outputs also in the text. (you should mention already in this section that you obtain the characteristic tradeoff
curves, given parameter bound and problem constraints)
(These are just a few points, but in general terms, the whole section should be carefully re-written)

We thank the reviewer for the critique of this section. Please see responses below and updates implemented throughout this section.

1. Please first introduce the model equations, then the objective functions and finally formulate the problem (1).

   We agree with the reviewer that the readability of this software note could always be improved. However, we are confused by this critique as the equations are already readily available. We have already provided the general problem formulation in the text. Next, the model equations (and constraint values) for the various test functions used in example 1 are presented in Table 1. Additionally, the model equations for example 2 have been previously published (and the reference given in the text). Lastly, the implementation of both examples is available in the JuPOETs repository on GitHub (the reference is also given in the text).

   However, we addressed the reviewer concern by updating the text of each example section with a description of the model equations, and more clearly pointed to either Table 1 or the associated reference for example 2. We have also updated the general description of the optimization problem that can be solved using the JuPOETs in the implementation section. Lastly, in each section we again reinforced that the implementation of each example is provided in the JuPOETs GitHub repository.

2. Please specify the type of model equations, type of decision variables (ODEs, real variables?)

   JuPOETs can handle ODEs and non-linear algebraic equations in real continuous variables as defined by Eqn 1. While the examples illustrate both of types of applications, estimating parameters that appear in systems of ODEs will likely be the most common use case. However, because the user controls both the objective function and the neighborhood/constraint functions, the basic run-loop implemented in JuPOETs could also be adapted to categorical, binary or mixed variable types.

   Toward this concern, we have updated the text in the implementation section to better explain what model and variable types could be used with JuPOETs and how the user could adapt the current run-loop to additional variable/model types.

3. Please define the trade-off surface

We thank the reviewer for this helpful suggestion. In the context of JuPOETs, the trade-off curve (or surface depending upon the number of objectives) represents the optimal balance between competing training objectives. Each point that lies along this curve represents the performance of a single parameter set. When lying directly on the curve, it is impossible to make the performance of one training objective better without making at least one of the other objectives worse. Thus, it is the best trade-off over all training objectives. We have updated the text with a better description of the trade-off surface.

4. Please elaborate more on the modified simulated annealing approach you use (please see comment 3)

The simulated annealing approach used by JuPOETs is the standard approach with the exception that the probability of accepting a parameter solution is based upon the Pareto rank of the solution instead of its error. We have already defined this in the text, and given functional forms for the acceptance probability.

Toward this concern, we have updated the text to make the connection with the tradeoff surface more clear to the reader. We have also standardized the use of technical terms (see point #5 below) and have added an original reference to the simulated annealing algorithm:

Kirkpatrick, S., Gelatt, C.D. Jr, Vecchi, M.P.: Optimization by simulated annealing. Science 220 (4598), 671–80 (1983). doi:10.1126/science.220.4598.671

5. Please provide definitions for all the technical terms you use (Pareto optimality, computational annealing temperature,…

We have updated the text to address these concerns.

6. Please describe the algorithm inputs and outputs also in the text. (you should mention already in this section that you obtain the characteristic tradeoff curves, given parameter bound and problem constraints)

We thank the reviewer for pointing put this important shortcoming. While the input/outputs of JuPOETs were already defined in Algorithm 1 and partially discussed in the text, we appreciate the critique that they could be better described in the text.

Toward this concern, we have updated the pseudo code in Algorithm 1 (and the associated caption) with a better description of the required inputs and expected outputs of JuPOETs. We also added an additional paragraph to the implementation section which describes the inputs, outputs and parameter values associated with JuPOETs (including discussing default behavior).

3. (CONCLUSIONS)

The following paragraph is misleading: "Many evolutionary approaches e.g., the non-dominated sorting genetic algorithm (NSGA) family of algorithms, have been adapted to solve multiobjective optimization problems [30, 31]. It is unclear if JuPOETs will perform as well as these other approaches; one potential advantage that JuPOETs may have is the local refinement step which temporarily reduces the problem to a single objective formulation. Previously, this hybrid approach led to better convergence on a proof-of-concept signal transduction model."

"It is unclear if JuPOETs will perform as well as these other approaches" this needs to be solved before publishing the method (otherwise, why not implementing NSGA instead?). I think that the problem is again related to a poor justification/contextualization of the tool. Hybrid optimization combines global and local optimization methods. The authors claim in the introduction and implementation sections that they use simulated annealing (which is global) without mentioning (at least explicitly or in a clear way) the combination with local search. Does JuPOETs implement a hybrid method? if yes, it should be specified in the implementation section. This would provide already an advantage over other methods (NSGA):

Hybrid methods are shown to perform very efficiently in previous works (in the context of biochemical systems)

http://bmcsystbiol.biomedcentral.com/articles/10.1186/s12918-014-0113-3

and examples (in the context of biochemical systems) are found in which hybrid solvers outperform pure evolutionary methods (NSGA-II), see:

http://pubs.acs.org/doi/pdf/10.1021/ie0605433

"one potential advantage that JuPOETs may have is the local refinement step which temporarily reduces the problem to a single objective formulation", the authors need to explain this with more detail already in the implementation section.

We thank the reviewer for pointing out this source of confusion. We compared JuPOETs to our previous POETs implementation in Octave. Since open source implementations of other non-linear/ensemble methods are not implemented in Julia, we are making a significant contribution by porting an updated flexible POETs implementation to Julia, and presenting this approach in a software note. Second, JuPOETs can be operated in “hybrid mode” where parameter space can be searched as a single objective problem. The output of this search can be used as the input to the multiobjective calculation. We have also shown previously (in the Song et al., 2010 POETs reference) that POETs run in “hybrid mode” out performs POETs without the single objective local refinement step.

To address this confusion, we edited the text to better reflect the innovation of the Julia implementation, we included the references suggested by the reviewer, and discussed the JuPOETs hybrid mode in more detail. We modified the pseudo code presented in Algorithm 1 to reflect where the single objective local refinement call occurs, we also modified the Implementation section to reflect the optional use of a hybrid mode with a local refinement step. Lastly, we changed the Julie implementation to publicly expose a user definable local refinement function pointer than can modified with whatever local search logic the user wants to implement. However, by default this function is a simple pass through function, thus, the user is required to implement local refinement logic only if they choose to do so.

Minor comments:

The paper needs careful proofreading (including some incomplete references, for example names are missing in ref 4).

We thank the reviewer for pointing out these typographical errors. We have corrected these errors throughout the text and in the references.

Reviewer #2:

The paper describes a software update of the one in references 24/25. From the technical point of view, it is fair but for my comments below. Yet, I do not think the new software version adds enough technical improvements to deserve a full paper. I would see it as a short note.

> We completely agree with the sentiments of the reviewer that this manuscript is appropriate for a software note. *The manuscript was originally submitted as a software note, not a full paper.* Its objectives were to describe a significantly updated POETs approach, make the community aware of JuPOETs (which we feel could be widely valuable), and more broadly to introduce Julia to the systems biology community. Thus, we feel this note is an important contribution to the systems biology literature. Toward this argument:
>
> *JuPOETs is simple to use, platform independent and open source*. While there are other multiobjective algorithms and implementations available, these are often not source controlled/distributed on GitHub, they require either manual C completion or a compatible platform to run a precompiled binary, or they require a costly commercial environment, such as MATLAB, to run. JuPOETs does not suffer from these disadvantages; it is simple to install into any Julia installation on any platform/system, it is distribution is centrally managed on GitHub and it is completely free and open source. Lastly, as the examples demonstrated it is also highly flexible. The learning curve to become productive in Julia is low (similar to MATLAB), thus, we feel researchers in the field can easily adapt JuPOETs to their problems.
>
> *Julia is a quickly growing high-performance computing platform that could be important to the community.* Julia is a relatively new general purpose computing language. However, it has experienced significant growth and adaption in the broader scientific community as a viable alternative to Matlab, Python and even C codes. In a recent InfoWorld article (*Languages on the rise: Julia, Go, Kotlin ... and assembly?* Paul Krill, 9/16/16) regarding the most recent Tiobe language rankings:
>
> "The Julia programming language is meant for numerical computing. It combines functional programming paradigms with high speed," a report accompanying the index states. "In other words, readable and stable code that performs. Chances are high that Julia will gain even more popularity the next few months."
>
> Tiobe Managing Director Paul Jansen said that Julia combines functional paradigms, readability, and speed. "In this way people can write code fast without sacrificing performance. There is no need to rewrite the prototype in another language." Jansen believes that Julia could become a top 20 language, challenging Python.
>
> We agree with this assessment. This software note will introduce the community to the advantages of Julia as a computing platform while simultaneously providing a useful tool for multiobjective optimization. Taken together, we feel, consistent with reviewer #2, that the JuPOETs manuscript warrants publication as a software note.

It is not clear in what particular sense the authors use the concept of ensemble modeling. From the examples shown, it seems that they refer to the possibility of considering fitness to different experimental data sets as different goals, thus producing a set of parameter sets along the Pareto front. Hence, different parameter sets are more or less optimal for different goals (exp. data sets), and an average set can be chosen to perform reasonable for all goals. While this idea is good, it is not a new one. The same can be used e.g. when there is lack of identifiability (see e.g. Villaverde etal. (2015) A consensus approach for estimating the predictive accuracy of dynamic models in biology. Computer Methods and Programs in Biomedicine 8:113) or when a reduced nonlinear model is to be fitted to different input signals that would require additional structure for

the corresponding outputs to be fitted in a single model, etc. In any case, the sense in which the authors are using the concept of ensemble modeling seems too forced, as strict ensemble modeling requires either (post)processing of the ensembles of models (e.g. Simidjievski etal (2016) Modeling Dynamic Systems with Efficient Ensembles of Process-Based Models. PLoS ONE 11(4)) or of the ones of parameters sets to give more than an adhoc solution lo lack of identifiability.

We thank reviewer #2 for highlighting important concerns about JuPOETs, and the manuscript in general. We address these concerns (each in turn) below:

1. **What do we mean by an ensemble?** We have updated the manuscript to better reflect what we (and many others whom we have referenced in the manuscript) mean by an ensemble; we are estimating families of model parameters (given a fixed model structure) that gives model performance consistent with experimental training data sets. It certainly would be interesting to simultaneously search model structures and parameters using JuPOETs, but this is beyond the current scope of this software note. However, we have published such a search previously, but the search did not use multiobjective optimization, see:

   Wayman J[#], Sagar A[#] and J. Varner (2015) Dynamic Modeling of Cell Free Biochemical Networks using Effective Kinetic Models. Processes, 3:138-160 (invited; *Special Issue Dynamic Approaches to Metabolic Modeling and Metabolic Engineering*)

2. **Each experimental data set as a different goal?** We agree with the reviewer; it makes intuitive sense for each training objective to be treated as a different goal. However, ultimately, the user partitions the training data into the specific training objectives; this is completely specified by the user in JuPOETs as they supply the logic of the objective function calculation.

3. **Identifiability?** We thank the reviewer for raising this important question. However, we respectfully disagree with the reviewer that lack of identifiability is somehow a shortcoming of JuPOETs. First, we have not made any claims about the identifiability of model parameters estimated using JuPOETs, and thus are unsure of the reviewer's concern. Instead, JuPOETs produces a parameter ensemble that is consistent with training or prediction data; it is focused on finding parameters that give good model performance rather than estimating the actual value of an identifiable parameter. Moreover, practical parameter identifiability is more of an experimental design question, rather than a parameter estimation problem. Previously, we have published techniques to estimate which parameters are identifiable using a Fisher Information Matrix (FIM) approach, and have recently used this approach to design a series of experiments to estimate the model parameters for RNA-based synthetic circuits. Second, a priori parameter identifiability is a function of the model structure and which measurements can be made. Again, this has nothing to do with JuPOETs. However, to make this point clearer, we have updated the manuscript with a disclaimer regarding parameter identifiability.

   Hu C[#], Varner J, and J. Lucks (2015) Generating effective models and parameters for RNA genetic circuits. ACS Synthetic Biol. 4:914-926 (available bioRxiv doi: http://dx.doi.org/10.1101/018358 and doi:10.1021/acssynbio.5b00077)

   Gadkar KG, Varner J, and Doyle FJ III. (2005) Model Identification of Signal Transduction Networks from Data Using a State Regulator Problem. IET Sys. Biol. 2:17-30

4. **Post-processing of an ensemble?** We thank the reviewer for pointing out this concern. The ensemble of parameters produced by JuPOETs does not require significant post-processing. JuPOETs produces an archive of parameters of different Pareto ranks that can be sampled directly to calculate model performance on training or prediction data sets. As shown by the second example, the overall archive contained parameter solutions that performed better or worse on

specific objectives, but collectively gave a consensus performance. To isolate sets for a specific objective one simply needs to sort the error archive. We believe this is an acceptable post-processing burden for the user.

However, there are cases where other minor post-processing steps may also be required. The most likely of these results from combining archives estimated from multiple initial parameter guesses. JuPOETs returns an archive of parameter solutions, parameter solution ranks, and the associated error of each solution in the parameter, rank and error archives, respectively. However, if JuPOETs is run from multiple starting locations, and the archives from each of these runs is combined into a single collective archive, the combined parameter rank archive may become invalid. In these cases, it is required to re-rank the parameters using the combined error archive. To facilitate this, the internal ranking function is exposed to the user. To make this point clearer, we have updated the text to make the user aware of this case, and to describe how to access the ranking function.