Revision 2021-19 entitled
A Review of R Neural Network Packages (with NNbenchmark): Accuracy and Ease of Use

Salsabila Mahdi, Akshaj Verma, Christophe Dutang, Patrice Kiener and John C. Nash

June 28, 2021

Dear Editor-in-Chief,

We are pleased to propose a revised version of the manuscript, *A Review of R Neural Network Packages (with NNbenchmark): Accuracy and Ease of Use* to R journal. We are grateful for the interesting and relevant comments of the referees.

Below, we detail our responses to points raised by referee #1 in Section 1, by referee #2 in Section 2 and by referee #3 in Section 3.

Yours sincerely

Christophe Dutang
the corresponding author

1 Referee #1

1. *Comments on Rubric 1: Utilities in R to deal with NN*

   a. predict function exists = 1 star

   b. scaling capabilities exist = 1 star

   1) Some packages tested received 0 star for scaling, but do support scaling through integration with other packages, like recipes, that provide frameworks for data preprocessing. In general, recipes could be used for scaling with any of the packages, and the packages do not necessarily deserve credit for the existence of recipes. However, some are specifically designed to integrate with recipes (or other preprocessing packages) by supporting model fitting calls of the following general form.

```
library(recipes)

rec <- recipe(formula, data) %>% step_normalize(all_predictors())

fit(rec, ...)
```

   In essence, recipes is a part of their interfaces. Recipes integration may require a bit more coding than built-in scaling, but does enable scaling and has the added advantage of enabling other types of preprocessing steps. A case could be made that capabilities are greater with the latter. Thus, integration with other data preprocessing packages warrants credit in the rating. To ensure that scaling capabilities are accurately characterized in the ratings, consider contacting the package maintainers to ask about their software ís support for scaling.

2. *Comments on Rubric 3: User-friendly call to fit a NN*
   a. simple one-line call or a single function = 2 star
   b. multiple-lines call to a single function = 1 star
   c. multiple-lines call to many function = 0 star

   A 2 star rating for nlsr seems unwarranted because its usage is more involved than other packages, like nnet, given the same ranking. In order to fit a NN with the nlsr::nlxb function, the mathematical form of the NN equation must be supplied as a formula. This requires more complete knowledge of the underlying NN than, say, nnet which requires specification of a formula only in terms of the response and a linear combination of the predictor variables. The operators (and functions) that appear in formulas are analogous to function calls. Accordingly, the formula in nlsr consists of more calls than the formula in nnet. Additionally, the testing code call to nlsr::nlxb is two lines and includes a call to the list function. Therefore, nlxb calls appear to be multiple-lines call to many function, which is more in line with 0 stars than the 2 star rating given.

3. *Rating packages on their ease-of-use is a worthwhile endeavor. However, some of the subjective components of the current rubric and its application detract from the ratings as a measure of ease-of-use. Consider the four examples below intended to illustrate how model fitting syntaxes might vary across different packages or users.*

```
# Example Syntax 1
fit(formula, data, model = "model_name", param1 = value1, param2 = value2)
# Example Syntax 2
model <- model_name(param1 = value1, param2 = value2)
fit(formula, data, model = model)
# Example Syntax 3
fit(formula, data, model = model_name(param1 = value1, param2 = value2))
# Example Syntax 4
fit(formula, data, params = list(param1 = value1, param2 = value2))
```

Syntax 1 is similar to that used in caret ***(fit = train, model = method)****, Syntaxes 2 and 3 to MachineShop, and Syntax 4 to* ***nlsr (fit = nlxb, params = control)****. The number of stars awarded seems to differ across these types of syntaxes, and the reason for the differences is unclear for a number of reasons. First, according to the current rubric, Syntax 3 might be considered less user-friendly than Syntax 1 because* ***model_name*** *appears as a second function call instead of as a character value. Staring the two differently would seem arbitrary given that they require the same knowledge and similar specification of the model and parameter names. Second, Syntaxes 2 and 3 differ only in the user is choice to define the model on a line separate from the fit call. The rubric to award more stars for fewer lines seems arbitrary in the case of these two syntaxes given that the number of lines is a user choice and not a package requirement. Third, Syntax 2, which is equivalent to Syntax 3, received a different number of stars than Syntax 4 even though both can be written in one line and consist of two function calls.*

*In summary, Rubric 3 has several subjective components. The term *simple* in item (a) is non-specific. The distinctions between single and multiple lines of code can be arbitrary in cases where the number of lines is a product of user choices. Single lines can often be written as multiple lines, and multiple lines as a single one (particularly with use of the %>% pipe operator). Additionally, there appear to be some inconsistencies in counting function calls (counting of* ***model_name*** *but not list) and arbitrariness in not counting function names supplied as argument values. Rubric 3 should be revised with more objective rules that are clearly described and consistently applied. For example, it makes sense to award fewer stars to packages that require a formula specification for the full NN equation (e.g.* ***nlsr::nlxb****) and more stars to those that only require a linear combination of the predictors in the formula or that accept x and y data structures directly. More thought should be given to the awarding of stars based on number of lines or function calls.*

2 Referee #2

Major issues

1. *The manuscript is not set appropriately in relation to the literature, e.g., the introduction has no references at all. Moreover, I think I found very few references, which are not R packages or datasets, and most of these are from the 1990s.*

2. *The restriction on single hidden-layer regression networks with tanh() activation does not reflect state-of-the-art neural networks used in practice. Although such a restriction is understandable from the author's practical perspective, it renders the comparison to be not very useful for applications of neural networks.*

3. *The description of neural networks on pages 2-3 is not up to date. For example, it is very common to use activation functions such as ReLU, which is not differentiable at 0 and not bounded.*

4. *For a benchmark paper, the number of included datasets is quite low, and the chosen datasets are very simple. Given that collections such as OpenML100 are readily available, a benchmark should be based on more and more complex data.*

5. *It is unclear how hyperparameters such as the learning rate were tuned. In a benchmark paper, hyperparameter tuning is essential to draw valid conclusions beyond the defaults.*

6. *The framing of this paper is not precise. The introduction states a still scientifically interesting hypothesis ("we hypothesize that these second-order algorithms would perform better than the first-order methods for datasets that fit in memory"), but according to the title, abstract, and evaluation criteria, they want to present a more general comparison of R packages.*

7. *Despite a very nicely designed and organized website (NNbenchmarkWeb), the documentation on GitHub is quite messy and without clear guidance on how to use the package.*

8. *Comparing the performance only on the training data is rather unrelated to common applications and does not indicate how well the network generalizes.*

Text-specific issues

1. *Figure 1: The figure caption does not describe the figure sufficiently well.*

2. *p. 2: It is unclear how Figure 1c relates to normalized inputs.*

3. *p. 2: Do not write large math equations inline. Use the display mode instead.*

Minor issues

1. *p. 1: "For regression and classification, the term multilayer perceptron is used interchangeably." - Multilayer perceptrons are a particular type of neural networks based on feedforward neural networks.*

2. *The term "Neural networks" is spelled in different ways: "Neural Network", "neural network", "neural-network", etc.*

3. *Sometimes quite drastic or fuzzy wording: p. 1 "[...] poor packages are implemented on CRAN.", p. 2 "[...] perform better than [...]" or p. 2 "[...] we believe it is helpful to have relatively large gradients [...]"*

Other remarks

1. *Regardless of the major issues in this paper mentioned above, the basic idea of reviewing existing R neural network packages is highly relevant in current research and should be pursued further.*

3 Referee #3

The minor changes are suggestions of aspects that can improve the manuscript:

1. *the writing could be improved in several cases, such as: removal of "oral" language: "there's" $->$ "there is";*

2. *change of title "Neural Networks: The Perceptron" $->$ "Multilayer Perceptron with a Single Hidden Layer";*
   in Table 1 "nb." $->$ size.

3. *the NN acronym in Fig.1 is not detailed, nor the notation 1-3-1 is explained.*
   Fig1 c) is not a single hidden layer networks.

4. *the examples in page 2 assume the tanh and atan activation functions, why not use $f()$, where f is the activation function, which can include the logistic function?*

5. *in Phase 2, it could be explained how the NNbenchmark was used, with one example (and its characteristics).*

6. *some figures, such as Fig2, should have a x-axis with numbers and labels.*

7. *it should be explained that is a first order and second order algorithm.*