Revision of our submission 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

October 14, 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.

In the pdf, changes for Referee #1 are put in red, for Referee #2 in blue and for Referee #3 in orange. Other changes are in cyan. Hypertext references are now used for tables and figures to ease the reading of the paper in pdf.

Yours sincerely

Christophe Dutang
the corresponding author

1 Referee #1

In the pdf, changes for Referee #1 are put in red.

1. *Comments on Rubric 1: Utilities in R to deal with NN; a. predict function exists = 1 star; b. scaling capabilities exist = 1 star*

   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 is support for scaling.

   We update the text to explain exactly how the utility rating was computed and also explain that many R packages provide preprocessing functions which can be used before the neural network fitting process. The `RWsearch` package lists 67 packages on CRAN to perform data preprocessing.

   ```
   > library(RWsearch)
   > crandb_down()
   > s_crandb("preprocessing", "data",
   +          select="TD", mode="and")
    [1] "bdpar"                "benthos"              "biclust"
    [4] "binst"                "bulletcp"             "ChIPtest"
    [7] "CITAN"                "clickR"               "cobalt"
   [10] "dataprep"             "discretization"       "ebal"
   ...
   [49] "rdwplus"              "recipes"              "RespirAnalyzer"
   [52] "RGCxGC"               "rminer"               "RobLoxBioC"
   [55] "shinyrecipes"         "sstModel"             "TDMR"
   [58] "torchaudio"           "torchvision"          "tosca"
   [61] "TSrepr"               "tsrobprep"            "vimpclust"
   [64] "VWPre"                "waves"                "wiseR"
   [67] "wvtool"
   ```

In the utility rating, we could have given stars for packages providing generic functions such as `print`, `plot`, `summary`, as it is supposed S3 objects are used. We did not contact the 25 package maintainers providing the 60 algorithms, as the description, the documentation (manual, vignettes,...) should be sufficiently clear and precise so that users find the `predict` function or the `scale` function without contacting the maintainer.

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.*

   Indeed, there are some inconsistencies with this rating and in particular `nlsr` was badly rated.

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.*

We rewrote the rating methodology as follows

a. **a single function with arguments passed as character, numeric, boolean or formula; and data as a data.frame or a matrix = 2 stars**
b. **a single function with model specification passed as a list or via a dedicated function; or data converted in a dedicated S3/S4 object = 1 star**
c. **multiple functions for initializing-converting-fitting = 0 star**

Hence the Call rating has been updated accordingly. We also split Table 2 into two different tables : the new Table 2 contains only the RMSE score and the time per `package:algorithm` **while the new Table 3 contains ease of use scores per package where additional columns have been added to clarify how the fitting function can be called.**

2 Referee #2

In the pdf, changes for Referee #2 are in blue.

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.*

   Appropriate literature review added in the introduction as well as new references to recent books added in Section 2.

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.*

   We updated the formal description of a neural network with a general activation function f.

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.*

   We add a new comment on that point

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.*

   We thank the referee for pointing out the OpenML100 database, now replaced by OpenML-CC18 database, which propose many datasets (3433) of which 1589 are for regression purposes. However, for our test purposes, we do not need such a large number of packages because our 12 datasets identify a large variety of situations where numerical difficulties occur for any training algorithm, such as nearly singular Jacobian, zero residual or divide by zero situations. Furthermore, the larger dataset `bWoodN1` has been tested on TOP10 packages (not only TOP5) We observe that only 2 packages (`CaDENCE` and `traineR`) have a RMSE minimum close to the RMSE of TOP5 packages. This suggests packages outside TOP10 will have a bad performance on `bWoodN1`. Finally, it would be an excessive task to perform an entirely new benchmark over 1589 datasets given our present computational and human resources, though we mention this possibility for future work in the conclusion. The Google Summer of Code has generously provided support to our two junior researchers (one for two parts of the project), but all other support has been from the authors' own resources.

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.*

   Due to the high number of package:algorithm pairs, we did not fully attempt to train all the hyperparameters of all packages. We justify this choice because of the tendency of users to use default hyperparameters. Moreover, not all packages offer the same hyperparameters. Based on our GSoc 2019 project, we considered `maxit` to be the most important parameter, as some packages performed rather differently despite having the same algorithm, sometimes even the same code for the algorithm, because they had different `maxit` values.

   It was harder to find a harmonized value of the learning rate value. Even for algorithms that were supposedly the same, at least in name, we seemed to need different learning rates for the same level of convergence. After a time-consuming grid search we chose a compromise between `maxit` and learning rate.

   As for the top 10 packages, especially the ones that basically stem from `nnet`'s BFGS, we tried to make sure all the other hyperparameters were harmonized as well. For those we didn't harmonize completely, such as `rminer` which had a difference of maximum allowable weights from the default `nnet`, we have included notes in the paper. Most of the trial & error was for first-order algorithms while for second-order algorithms `maxit` is uniquely set to 200. We added a comment on this in the paper.

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.*

   We clarify this point by adding a paragraph in the introduction.

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.*

   An update of the project website has been carried out to reorganized notebooks, results. The website is now hosted at `https://theairbend3r.github.io/NNbenchmarkWeb/index.html`.

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

   We only use a training set because the purpose of our study is to verify the ability to reach good minima, i.e., rating optimization methods used in NN packages. This requirement is satisfied by using only a training set.

Text-specific issues

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

We clarify the NN *a*-*b*-*c* notation and also add words in the captions

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

    We clarify this point in the text

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

    Done

Minor issues

12. *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.*

    TODO Patrice?

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

    Done

14. *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 [...]"*

    Done John, could you validate?

Other remarks

15. *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.* **We are grateful to the referee for this comment.**

3 Referee #3

In the pdf, changes for Referee #3 are in orange.

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";*

   I Ctrl+Find 's in the paper. John, could you validate?

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

   Done: I do not agree with Table 1 : parameter size is not correct to me John

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.*

   We formally introduce the NN a-b-c notation.

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?*

   Yes, we generalize the NN formulation in order to have a single equation.

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

   We add comments on this part, but the full example is left in Appendix C.

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

   Done

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

   Done, we add two sentences in the introduction.