We would like to thank the anonymous reviewer for the helpful comments concerning our article. We try to do our best to improve the paper and package, and to correct all of their flaws. Some spelling errors in the paper were corrected, and bugs in the package were fixed. To simplify the review process, we provide the additional pdf file with the marked differences between the old and the new version of our paper. Please also find the detailed answers to the reviewer's remarks below:

1. *missing unit testing - there exist several very good packages for unit testing of R packages, see e.g. the "testthat" package. In my opinion, it is necessary to provide good testing suite for any software, especially the mathematical functions. Not only the main functionality must be tested, but also the border or error cases such as empty inputs, NAs, NULLs etc.*

The respective tests (based on the above-mentioned "testthat" package) were added (like expect_snapshot and expect_equal for the expected correct output, together with expect_error to check the behavior of the functions in the case of the introduced errors).

2. *inconsistent naming conventions for functions - it is a problem in R that there is not a standard for the creation of function names. Some users prefer lower_case_function_names, some use camelCaseNames or CamelCaseNames. It is important to at least follow the selected style, which is unfortunately not the case in this package: some functions follow the CamelStyle with first letter being upper-case (see function "BertoluzzaDistance"), but some other functions start with lower-case ("classicalBootstrap"), some function names even violate the selected CamelStyle at all ("wmethod", "dmethod").*

Names of the functions were corrected to follow CamelCaseNames style. Only for the vector of names (resamplingNames, samplingGenerators) camelCaseNames style was used to clearly distinguish these vectors from the functions.

3. *assertions on input values are insufficient - a good package should test thoroughly the validity of inputs. For instance, BertoluzzaDistance() should test explicitly that only numeric vectors or matrices are allowed as inputs. Unfortunatelly, putting e.g. a 3D array as the input causes the function to fail in an ugly way: BertoluzzaDistance(1:5, array(1:12, dim=c(2,3,2))) Other function suffer from similar problems.*

In the functions BertoluzzaDistance, ParameterCheckForInitialSample (which is used to check initial samples for other functions), and many others, additional conditions were applied (e.g., is.matrix). Together with the previously used conditions (if the number of columns is equal to 4, is.na, is.numeric, etc.), we hope that the most common errors in the validity of input parameters can be directly reported to the user. Some new conditions (e.g., if the increases parameter is a logical one) regarding parameters in other functions were also incorporated to better handle the possible errors.

4. *I would suggest to introduce an S3 class for trapezoidal fuzzy numbers and provide basic methods for working with such class - e.g. transformation from/to matrix, selection, printing, etc. The S3 class could also handle both types of trapezoid representation, so that the "increases" argument would not be needed in a lot of functions. The package could then act as a basic tool for working with trapezoidal fuzzy numbers.*

We would like to thank the reviewer for this interesting idea. Our goal regarding this package is to focus on providing the resampling methods together with their possible applications (e.g., in statistical analysis) and to keep this package as concise as possible. There are other very helpful and advanced packages that are aimed at working with trapezoidal (and other) fuzzy numbers (e.g., FuzzyNumbers package). We have some plans related to functions that can be used to "transfer our fuzzy numbers" into the "language" of other packages (like the above-mentioned FuzzyNumbers), but we will deal with them in the future.

5. *It would be nice if the generators of random fuzzy numbers allowed an arbitrary distribution to be selected. That would increase the usability.*

The general function GeneratorFuzzyNumbers was added. This function can be invoked using various random number generators (e.g., rnorm, runif) from the stats package. The specialized functions GeneratorNU and GeneratorNExpUU are now written as the special cases of this universal function.

6. *use of "lapply" or "vapply" is more efficient than "for" loops (as. e.g. in OneSampleCTest)*

The reviewer is of course absolutely right. In the package, we try to balance possible approaches. We decided to use "for" loops instead of "apply" functions, because (from our point of view, e.g., in the case of OneSampleTest), "for" loop is easier to understand by the users, the variable "boostrapSample" is the whole matrix (not a single vector) which is then used to calculate "bootstrappedStatistics[i]", and the variable "numberOfSamples" can have big value (so allocation of additional memory for the new bootstrapped samples is important, the same also applies to the size of the bootstrapped sample). Please also note that some memory pre-allocation was introduced (e.g., for the whole vector "bootstrappedStatistics") to increase the numerical effectiveness of the function.

7. *In the introduction, the authors enumerate various characteristics of fuzzy numbers such as value, ambiguity, expected value etc. It would be beneficial to provide functions in the package that can compute that features.*

Now all the functions related to the calculation of the various characteristics of fuzzy numbers are directly accessible to the users (e.g., CalculateValue).

8. *Section "Estimation of SE/MSE", page 5, eq. 4: it is unclear to me, how do you sum the trapezoidal fuzzy numbers?*

The respective formula for trapezoidal fuzzy numbers was given in Introduction (p. 2/3):

“To introduce basic arithmetic operations (such as addition, subtraction, etc.) for the fuzzy numbers, the Zadeh extension principle should be applied (see, e.g., \citep{ban_coroianu_pg}).

However, in the case of TPFNs, some operations are easier to conduct, e.g., for $A = (a_1, a_2, a_3, a_4) $ and $B = (b_1, b_2, b_3, b_4)$, we have

\begin{equation}

A + B = (a_1+b_1, a_2+b_2, a_3+b_3,a_4+b_4) ,

\end{equation}

so the result is also a TPFN.”

9. *Section "Comparison of the resampling methods..." - this section made me believe that the functions "ComparisonSEMean", "ComparisonOneSampleCTest" and "ComparePowerOneSampleCTest" are included in the package, which is not the case. I do not see much usefulness to read about functions that are not accessible by the user from the described R package.*

All of the mentioned functions are now included in the package and are accessible by the users. Only the function CompareRealCaseTwoSample is given in the external (example) file because it is related to the special real-life sets (also provided in the same file) and is a simple variation of TwoSampleCTest function.

10. *Test size estimation - it would be interesting to see also the results for different significance values. Perhaps a Q-Q plot can be generated to show the correspondence of empirical and theoretical distributions of p-values.*

The additional plots (see Fig. 1 and Fig. 2) of the estimated percentages of rejections as a function of significance value were introduced to illustrate this relation. The respective sentences concerning these new plots were also added. We think that Q-Q plot may not be the best way to show these outcomes because of the problems with the theoretical distribution function for test statistics in the case of the one-sample C-test.