*The R Journal* – **Submission 2019-80**

Mario Gonzalez Sales

Address: Ctra. Engolasters s/n

AD700 Escaldes-Engordany, Andorra

Dr Michael Kane

Executive Editor

The R Journal

November, 29th, 2019

Dear Dr Michael Kane,

Thank you for your valuable comments and suggestions to the manuscript entitled: "Assembling Pharmacometrics Datasets in R: the puzzle package". In response to the points raised by the reviewer, we would like to offer the following answers.

**Comment #1:** I had an opportunity to take a look. In general the package has potential to be useful to the pharmacometrics / data science community. However, the implementation of the package can be improved in various ways, and is not fully up-to-date with current best-practices (see below). I would suggest to the authors to first go through a CRAN submission to help them get the package up to current standards.

**Reply to comment #1:** We understand the reviewer concern. The puzzle package is now available on CRAN.

**Comment #2:** The manuscript itself gives a clear overview of functionality. It can be improved at various points in language, although in general it is definitely clear what is meant to the target audience. It could benefit from a bit more technical details, e.g. what libraries are used under the hood and how certain data types are handled.

**Reply to comment #2:** We concur. We have updated the manuscript improving the language points raised by the reviewer. Moreover, we have added more details that are technical within the manuscript to add clarity.

**Comment #3:** “time in- and/or dependent covariates” —> “time dependent and/or independent”

**Reply to comment #3:** The text has been modified as suggested.

**Comment #4:** “skyrocket the complexity …”. I agree that this makes it complex, but what I think is even more important is to highlight that it makes the process hard to script up-front, i.e. it will require different programming for each new analysis.

**Reply to comment #4:** The text has been clarified as suggested.

**Comment #5:** “.. the NONMEM software”. It might be good to highlight here or in the discussion that various other modeling packages (Monolix, nlmixr, mrgsolve) are also mostly compatible with NONMEM data structures, so the prospective user group would not be limited to NONMEM users.

**Reply to comment #5:** The text has been clarified as suggested.

**Comment #6:** “limitations and inconvenient of this framework” —> “limitations and inconveniences of this framework”?

**Reply to comment #6:** The text has been modified as suggested.

**Comment #7:** Section “The puzzle function”: I think before you dive into the various arguments, I think it would make sense to first show a simple example of a puzzle() call, e.g. with just pk, dose, and covariates. It then becomes more obvious to the reader that “pk” and “pd” are arguments to the puzzle function, and not R statements on their own.

**Reply to comment #7:** We thank the reviewer for raising this point. We have arranged the sections of the manuscript as suggested. In particular, section 2 illustrates two simple examples of data assembling using the puzzle package (i.e. one for “pk” and another for “pd”). Section 3 gives an overview of the arguments of the puzzle() function. Section 4 presents the pre-formatting requirements of puzzle(), and section 5 concludes with the discussion.

**Comment #8:** “Moreover, each sheet has to have the same items, and the items have to be in the same order.”. Please explain what is meant with “items”, I assume the columns in Excel?

**Reply to comment #8:** Yes. In NONMEM jargon, each column or variable is called “item” while each row is known as event or “record”. This is explained in the introduction section: *“Each row represents a record or an event, while each column represents an item or a variable”*.

**Comment #9:** character values: please expand a bit here on how factorial data is handled: categorical data should in principle not be coded as “character” within R (although it often is) but as “factor” data. Will the puzzle function also work with factor data? When the data is “character”, will it convert the character strings first to factors and then to numbers?

**Reply to comment #9:** Yes and yes.The puzzle() function works also with factors. When the data is a character, each string will be converted into a different factor and then, each factor will be coerced into a numeric value. This is because NONMEM only accepts numbers and not characters. For example, let’s assume we have a character variable accounting for renal impairment (e.g. RENIMP) with the following possible values: “mild”, “moderate” and “severe”. The puzzle() function will convert the variable RENIMP from character to factor. Each string will be a different level. Then, the factor variable will be coerced into numeric. If there is no numeric order within the factor, the levels will be alphabetically assigned. Moreover, puzzle will keep track of all coerced variables with a message printed in the console and with an external file if a name is provided with the coercion argument. This has been clarified in the manuscript.

**Comment #10:** One danger with categorical data is that e.g. a new dataset contains a new categorical level that wasn’t present in earlier datasets. This could then change the numbering. Is there anything the user can do to avoid this or be alerted to it?

**Reply to comment #10:** Yes, the user will be alerted in two ways. On one hand, with the message printed by puzzle() in the console, and on the other hand, with the coercion file, generated if specified with the coercion argument.

**Comment #11:** “structure diverse” —> “structurally diverse”

**Reply to comment #11:** The text has been modified as suggested.

**Comment #12:** Section “Pre-formatting requirements”: I think the authors are correct in that pre-formatting might also take a long time, potentially longer than actually combining the data. That’s why I see the strength of the puzzle() package primarily in automatically documenting the data assembly, and unification of the assembly workflow and output data files across a team. I don’t think that, especially for R users with a lot of tidyverse experience, puzzle will make a huge improvement in speed of data assembly once pre-formatted data is available. However, it is the factors I mentioned above that do make it potentially very interesting for pharmacometrics groups. I think this can be highlighted more in the Conclusion as well.

**Reply to comment #12:** We concur. We strongly believe that the puzzle package fills an unmet condition within the pharmacometrics workflow as it offers an opportunity for a unification of the code for data assembly improving reproducibility and traceability. We have highlighted these factors in the conclusion.

**Comment #13:** Also, given the above, I think that especially for industry users it is important that puzzle documents all steps, transformations, warnings, etc. I would therefore suggest, perhaps optionally to the user, to add more printing / logging to the console. From the output shown for Example1 it seems only covariate coding is shown. I would add information and statistics like assumed version of puzzle package, timezone, number of patients found, number of observations, number of missing PK data, username of person doing the assembly, etc.

**Reply to comment #13:** The output showed in Example 1 corresponds to a very simple example. The puzzle() function outputs the compartments assigned to each analytes in addition to the variables coerced to numeric. Moreover, we have added additional information as the reviewer suggested. The puzzle function also outputs the number of subjects and observations. The date and time of the data assembling, the timezone, the different dose levels administered, and the person who assembled the data set if defined with the argument username.

**Comment #14:** preformatting requirements for dosing records: would suggest to add some guidance here on how to add infusion rate, as that is a common feature. I would assume it is as easy as adding a RATE column in the dataset?

**Reply to comment #14:** Correct! If the file containing the dose information (e.g. dose.csv) has also the variable RATE, and order is set to 0, the pharmacometrics data set will be assembled assuming an infusion administration. In this regard, a clarification has been added in the manuscript.

**Comment #15:** “databsets” —> “datasets”

**Reply to comment #15:** The text has been modified as suggested.

**Comment #16:** “the main inconvenient of puzzle()” —> “inconvenience”.

**Reply to comment #16:** The text has been modified as suggested.

**Comment #17:** “this feature what” —> “this feature that”

**Reply to comment #17:** The text has been modified as suggested.

**Comment #18:** “code and storage the data” --> “store the data”

**Reply to comment #18:** The text has been modified as suggested.

**Comment #19:** The authors should mention that SDTM is a CDISC standard and refer to their documentation.

**Reply to comment #19:** We have modified the manuscript according with the reviewer suggestion.

**Comment #20:** “may evolve into a simple breeze” —> I believe the expression is “be a breeze” or perhaps “become a breeze”.

**Reply to comment #20:** The text has been modified as suggested.

**Comment #21:** If the puzzle package is not available on CRAN, the authors should add a link to their Github repo. I would however advise to first publish to CRAN, then publish this article.

**Reply to comment #21:** The puzzle package is now also available in CRAN.

**Comment #22:** it seems RSQLite should be added as dependency in the DESCRIPTION file? I’m getting an error that it was not installed.

**Reply to comment #22:** We used the function sqldf() from the package sqldf. The sqldf package depends on RSQLite and this is the reason why RSQLite needs to be installed.

**Comment #23:** Why is RSQLite used as dependency? If the package is built on top of an SQLite database it would be good to mention that in the manuscript. In general it might be good to understand a bit better the technical details on what is used underneath the hood (e.g. tidy verse packages or base R etc), please add to manuscript.

**Reply to comment #23:** As mentioned in reply to comment #22, we used the sqldf() from the package sqldf which depends on RSQLite. We have added more technical details on what is used under the hood in the manuscript.

**Comment #24:** When just running the command “puzzle()” I’m getting an error (“Error in regmatches(x, regexec("\\.([^\\.]+$)", x))[[1]] : subscript out of bounds”). I would suggest to add a friendly warning or pointer to the help-files.

**Reply to comment #24:** The puzzle code has been modified as suggested by the reviewer. If a user runs the “puzzle()” command, the following informative error will be shown:

Error in puzzle() :

Have you forgotten to define the arguments for puzzle()?.

Do you need help? Please visit: https://syneoshealth.github.io/puzzle

**Comment #25:** It is bad practice to use `library` inside package functions, e.g.:

`suppressPackageStartupMessages(library(“readxl”))`. Instead use specific pointers “readxl::read\_xlsx()” and include the required libraries in the DESCRIPTION file.

**Reply to comment #25:** Internal code has been updated as the reviewer suggested. There not library calls. Moreover, specific functions from external packages are called using “::”.

**Comment #26:** the package has no tests. Before it can be submitted to CRAN or advised for use in production it should definitely include a well rounded test set, especially given the importance in the data workflow.

**Reply to comment #26:** We have performed extensive testing ending up with 0 error, 0 warnings and 0 notes. In addition, the package is available on CRAN.

**Comment #27:** it is good practice to split out the functions into multiple files so development and debugging is much easier. For functions that do not need to be exported just don’t include the @export tag.

**Reply to comment #27:** We understand the reviewer concern. Although we agree that split out the functions into multiple files is helpful for debugging we prefer to keep some functions under the hood to clearly differentiate which functions the user needs to learn. In this regard, we have performed extensive testing and we want to make the user’s experience painless. Therefore, we believe that it will be easier for users if they only focus in the function of their interest (i.e. puzzle()) rather than being tempted to explore other features of the packages from where they will not benefit.