Response to Reviewers

Eric Hare

04/10/2015

1. *It would be best to avoid using . in the names of functions except when defining methods. In particular, make.RV() is badly named, since "RV" is the class of the main objects created in this package, so this looks like a method for a make() function. RV() would be a better name. (as.RV() is less bad since this is a common paradigm in R and it is unlikely that anyone will define a generic names as().)*

   We have renamed the `make.RV` function to `RV`, and updated the paper accordingly.

2. *I think joint() and iid() would be a better names than mult() and multN(). (Also, the description of what these do in Table 1 is incorrect. They return random variables, not pmfs.)*

   We have renamed the `mult` and `multN` functions to `joint` and `iid` respectively, and updated the paper accordingly. We have also clarified the descriptions of these functions in the table.

3. *Even better would be to overload the + and* operators and use A + B for sofI() and A * B for mult()*

   The `+` and `*` operators are now overloaded. One may call `X + Y` to form the sum of two independent random variables. Similar functionality now exists for joint distributions, where you may call `X * Y` to form the joint of X and Y. Note that if `X * X` is called, it will simplify return the variable obtained by squaring each outcome, and not the joint of two independent realizations of X, so that computations like `E(X^2)` still make sense.

4. *In the examples, I would prefer that the names of all random variables be capitalized since this is a common notation to distinguish between random variables and the values they can obtain used in many text books.*

   All random variables used as examples in both the documentation of discreteRV and in the paper now begin with a capital letter.

5. *For random variables, I prefer value to outcome as a way of naming things. One could force the values to be numeric so that only true random variables can be produced. As is, we get the following psuedo-random variable:*

   We did consider enforcing that random variables have numeric outcomes, as mathematically they must. But we did away with this requirement so that the use cases for discreteRV could be expanded to realms where it is easier to refer to factor outcomes. Of course, this greatly reduces the functionality of the package as pointed out. We have updated the paper to clarify that numeric values are preferred for outcomes, not strictly integers.

6. *Random variables with infinite support: This is a bit tricky, and the current implementation is not entirely successful.*

   We have attempted to improve the support for random variables with an infinite number of outcomes. This is not perfect in every instance right now, and more work is being done. However, it will handle the cases listed in the paper and be more stable in general.

7. *Since "RV" objects are just vectors with attributes, they inherit behaviors from vectors, some of which are desireable, but others not*

   The undesirable inherited properties of vectors pointed out have now been solved by overloading the + and * operators as described above.

8. *If aimed at beginners, it would be best if all objects had a custom print() method to avoid ugly displays with attributes showing like this (copied from the manuscript)*

   A print method for an `RVsim` object has been added.

9. *Also, it is probably better to reverse the order of the arguments to rsim().*

   The arguments to `rsim` have been reversed, as suggested.

10. *Each "RV" object appears to store the outcomes three times. It is unlcear that there is any advantage to this design, and it is potentially error-prone if some operations don't update all three places.*

    We agree that storing the outcomes multiple times is an issue. We are striving towards removing this limitation. The outcomes attribute is only used in the case of a random variable with infinite support, to store the bounds for printing purposes. The other two are used, one to make syntax more friendly and the other is stored numerically to allow for the precision needed. We have corrected issues with, for instance, `X^2` not updating these outcomes properly, and will work towards a better future solution which isn't as likely to produce issues.

11. *A nice addition would be a simple way to create random variables with common distributions without having to write a function defining the probability mass function.*

    We have added support for three familiar distributions, which are documented in the `RV` function documentation. For instance, to construct a poisson random variable with mean parameter 3, you would call `RV("poisson", lambda = 3)` We will work on adding support for more in the future.

12. *If the package could be designed to handle joint distributions well, that would be a significant strenthening of the package.*

    Extensive work has been done with regards to joint distributions and the power of discreteRV in this realm:

    - **A simple interface for defining a joint random variable has been created. Outcomes can be specifed as a list of length n, where n is the number of variables involved. For instance, to create the variable XandY used in the review example, one would call `XandY <- jointRV(list(1:3, 0:2), 1:9/sum(1:9))`.**
    - **Marginal distributions now keep reference to their joint distribution, allowing for the computations explained in the review. To derive the marginals, one would call `X <- marginal(XandY, 1)`, and `Y <- marginal(XandY, 2)`. Then probability statements such as `P(X == 1 | Y > 1)` can be made. Further, `X | (Y > 1)` will return the random variable obtained. The functionality described in the review has been implemented, and it should support more than two random variables, although there are certainly edge cases that likely will need to be addressed in future updates.**

13. *In particular, creating winning outcomes using paste() is awkward. Something like the following would be better if could be acheived.*

    The statements in the extended example have been revised according to the review. In particular, outcomes are no longer generated using paste, but instead by exploiting the new joint random variable functionality.

14. *I was a bit curious about the simulation at the end of the article: the authors chose to perform the simulation by creating 1000 replication of 100 replications of the craps game, when the goal of the simulation could have been attained simply by 100,000 replications of the game.*

    We agree that the simpler notation for the craps game simulation is preferred. We initially had constructed the simulation differently, and did not update the code properly. It has now been updated accordingly.