We thank the reviewers for their detailed comments about our paper "Regular expressions and reshaping using data tables and the nc package." We have revised our paper in consequence, as explained below.

Reviewer 1

While this paper does an excellent job of placing the work under discussion in the context of existing relevant packages and approaches in R, it is this reviewer's opinion that this paper is two papers conflated into one: (Case 1): it's either a paper about name-capturing regular expressions OR (Case 2): it's a paper about (one direction of) data reshaping. The paper's clarity suffers because of this.
It is this reviewer's opinion that since the author's contribution to name-capturing regular expressions in R has been previously documented in their previous R Journal paper, case 2 is more likely to be of "new" interest to the R community...
2) In the second case (a discussion of data reshaping approaches), the sections "Uniform interface to three regex engines" and "Data table integration and nc::field to avoid repetition" should be dropped. A reference to the author's previous R Journal paper, along with references/links to appropriate package vignettes (I see passages called "nc::field for reducing repetition" and "Choice of regex engine" on the nc github repository README).
Response: we have adopted the suggestions based on case 2, by removing those two sections, and adding references to the package vignettes.

- Because the terminology that the author uses (reshape columns, copy columns, capture columns) is novel and unfamiliar to the user, it would be helpful to slow it down a bit; perhaps demonstrate the terminology (and include the relevant figures) closer to the examples of capture_melt_single and capture_melt_multiple.
Response: the new terminology has been used in the sentences/paragraphs immediately surrounding the examples of capture_melt_single and capture_melt_multiple. For example "The output data table consists of one copy column (Species), two capture columns (part , dim), and a single reshape column (cm)."

- It would also be helpful to do comparisons of (representative) packages on the same (simple) data tasks (like iris), to clarify the differences in design philosophy/approach, and to presumably highlight nc package advantages. Obviously when packages have highly similar interfaces, like tidyr and reshape2, doing both isn't necessary. This would also serve as another opportunity to familiarize the reader with the author's reshaping terminology, by applying it to reshaping packages that the reader is presumably more familiar with.
So, for example: "all" the packages on the iris examples and all the applicable packages on the treatment table example, or some other example suitable for highlighting the differences/advantages of the nc package's approach.
Response: data.table and tidyr code has been added for the iris examples.

- I would prefer to clarify the scope of the contribution earlier in the paper, near the top: namely that the paper only addresses one direction of data shaping (and why). I know that this clarification is in the abstract, but I confess I read the entire paper wondering

when they would address the opposite direction; and then why they didn't (not necessary to use regexp? future work? not a common use case?).
Response: we have added the sentence in the introduction, "Note that in this article we do not discuss tall-to-wide data reshaping, because regular expressions are not useful in that case." We have also emphasized that only one direction of reshaping is examined by changing the title to "Wide-to-tall data reshaping using regular expressions and the nc package."

And of course the use of regular expressions isn't helpful when the column names ("copy columns?") of the original data don't adhere to a regular naming convention.
Response: in the introduction we have clarified that point by adding a parenthetical at the end of the sentence "A main thesis of this article is that regular expressions can greatly simplify the code required to specify wide-to-tall data reshaping operations (when the input columns adhere to a regular naming convention)." In the abstract we have clarified this point by changing the parentheticals in "Capturing regular expressions that describe information to extract from column names can be especially useful when reshaping a data table from wide (few rows with many regularly named columns) to tall (fewer columns with more rows)."

Reviewer 2

The syntax is very R-like in that it uses named arguments. This approach breaks down a bit with the fields() function introduced by the author to reduce repetition. I would suggest the named argument syntax is to be preferred despite the repetition, because it makes it very clear what columns are created in the output.
Response: thanks for the suggestion about clarity. The section which discussed the field() function has been removed.

The reshaping functions return a data.table by default. This choice fits the spirit of the nc package which is focused on a terse syntax. However I would recommend using a default output type that behaves more like a data.frame , in the interest of using standard semantics shared by the R community. Data.tables share the same data structure as data frames but implement a substantially different interface (unlike tibbles which mostly behave like data frames).
Response: The choice of data table output is now explained via the new sentence "The output above is a data table (a data frame subclass with special methods that have reference semantics) because data.table::melt is used internally for the reshape operation."

For these reasons, we have started switching all the examples in our packages from iris to the penguins dataset which was designed to support the same sort of examples: https://github.com/allisonhorst/palmerpenguins
Response: You are right that Ronald Fisher's racism was and still is problematic, and thanks for the suggestion to use the penguins data set instead of iris. The penguins data set is an excellent example of "missing input reshape columns" so I added an example which uses the penguins data to the capture_melt_multiple man page in this development branch, https://github.com/tdhock/nc/pull/12 However I would like to keep the iris data set as an easy to understand example in the paper (e.g., Fig 1) because it has no missing input reshape columns.

• The article uses non-standard tidyr idioms. For instance pivot_longer(data, cols = grep(re, names(data))) instead of pivot_longer(data, cols = matches(re)). I recommend using standard idioms.
Response: grep has been changed to matches in two places in the paper.

• The author mentions several times that pivot_longer() does not support arbitrary transformations. This is no longer true since the 1.1 version (May 2020) which introduced the transform argument. I recommend updating the manuscript to take into account this new feature.
Response: thanks for the suggestion, the paper has been updated.

If a benchmarking section is included, it would be better to be technically accurate regarding what is measured. Currently the measures mix the regexp engine and the reshaping engine (which I expect to take the bulk of the time), and the prose is not very clear about this.
Response: we have added a sentence to the experimental section to clarify this point, "Note that these timings include both the regex matching (which should be relatively fast) and the data reshaping operation (which should be relatively slow)."

I would really appreciate if the author updated the benchmark where nc takes 15s and tidyr takes 1 hour. They reported this to us on October 29, 2019, and the performance bug was already fixed in the development version of rlang, which has long been released to CRAN.
Response: the benchmark has been updated and tidyr is now much faster in the figure (comparable with other packages, even for large data sizes).