Reviewer A:

1. The point about the shortcomings of kinship2 not representing cross-generational breeding is accurate, but it does handle some simpler instances of this issue by making a complete copy of the relevant individuals linking them with a line (and their id) if the method is unable to align them with only one instance. I suggest a single sentence or clause that acknowledges this.

We added a paragraph in Section 2 that acknowledges this:

"In the kinship2 package, if an individual cannot be represented with only one instance, then it will be completely copied and connected with dotted lines to the relevant individuals. However, the package requires that each child has exactly zero or two parents; if a child has two parents, then one must be female and one must be male. These requirements preclude certain genealogical datasets from being plotted by generation count, especially when their complexity increases with inter-generational breeding."

2. The first paragraph of section 11 reads too much like a children's book narrative. The first sentence could be removed, and maybe just start the second sentence with "An academic genealogy is the second dataset provided in the package, where every parent ..., as briefly described in section 4".

We changed the first two sentences of the first paragraph of Section 11 to read as follows:

"An academic genealogy is the second dataset provided in the package, where every parent is also a child and some children have more than two parents, as was briefly described in Section 4."

3. The description of figure 8 has "two varieties of interest", which sounds like it is applying to the soybean data, and maybe should state "two statisticians of interest".

We changed the sentence before Figure 8 to read as follows:

"We set the fontFace variable of the plotPath() to a value of 4, indicating we wish to boldface and italicize the two statisticians of interest."

4. My only concern is that I think the paper lacks some applications that may apply to some quantitative features of the genealogical data, which you touch on in the first paragraph of the Introduction, but never get back to. For example, if you could query to get the varieties with the highest yield (a variable in sbGeneal), or query the statisticians with a "Rank" or "F[f]actorial" in their thesis, that would be a start. But furthermore, there could be a tests of which soybean branch (compared against sub-branches) seems to have the best of a certain feature, like yield. This kind of calculation would have applications in many areas.

We extended to the code and provided new examples in the paper to illustrate the use of different quantitative variables onto the horizontal axis of the plotPath() and plotPathOnAll() functions. The outputs can be seen now in Figures 4-Left and 7-Left. We also provided new examples of hard coding two quantitative variable (one for each axis) in the plotPath() and plotPathOnAll() functions. The outputs can be seen now in Figure 4-Right and 7-Right. For these examples, we not only show the figure outputs, but we also briefly explain with both text and code.

In addition, we added two new functions in the package called getBranchQuant() and getBranchQual() that can perform parsing and basic calculations on a variable of interest across the descending branches of an individual of interest. The variable of interest can be quantitative or qualitative respectively. We added a new section in the manuscript called "Branch parsing and calculations", where we provide code examples that demonstrate these two new functions.

Since the genealogical structure must be input as a data frame, base R data frame manipulation functions can be used to query certain features in the dataset. For instance, if a user wanted to determine which varieties had the largest yield, they could do this without any ggenealogy functions. Instead, they can simply sort the input genealogy data frame by the variable of interest:

```
R> install.packages("ggenealogy")
R> library("ggenealogy")
R> data("sbGeneal")
R> sbGeneal[with(sbGeneal, order(-yield)), ]
```

However, sometimes users may wish to query for a variable, only now also taking into account parent-child relationships. The ggenealogy functions can make this possible. As a result, the examples in the new section of the manuscript demonstrate that users can quickly and flexibly parse descendant branches. The swiftness comes from ggenealogy functions that allow for fast parent-child traversals, such as getChild(), getDescendants(), getBranchQuant(), and getBranchQual(). The flexibility comes from data frame manipulation functions in base R (such as matching and sorting) that can be used in conjunction with the parent-child traversal methods.

Reviewer B:

1. On page 3, a comparison is made with the package kinship2, and it is mentioned that this package can not unambiguously represent intergenerational crosses, and that is true as you show in Figure 1. However, does this package allow to represent not-standard genealogies, with a variable number of Parents? Maybe this should also be clearified.

We added a paragraph in Section 2 that discusses more about package requirements:

"In the kinship2 package, if an individual cannot be represented with only one instance, then it will be completely copied and connected with dotted lines to the relevant individuals. However, the package requires that each child has exactly zero or two parents; if a child has two parents, then one must be female and one must be male. These requirements preclude certain genealogical datasets from being plotted by generation count, especially when their complexity increases with inter-generational breeding."

2. On page 7, section 5, the input format of the package is described. As a suggestion, I would suggest to change variable "year" more general to "date", since you then also allow for shorter generation intervals. This might be the case in genealogies of insects, and other species.

We made several changes to the package syntax to fully address this recommendation:

a) We changed the variable label "year" to instead be a variable label that the user can specify as any character string column name for a quantitative variable in the dataset.
b) We also changed the method name "getYear()" to "getVariable()".
c) We changed the list returned from the "getPath()" method to contain objects "pathVertices" and "variableVertices" (as opposed to "pathVertices" and "yearVertices").
d) We changed the default horizontal axis label for the plotPath() method to be the column name of the quantitative variable in the dataset specified by the user instead of "Year".
e) We changed the default horizontal axis label for the plotPathOnAll() method to be the column name of the quantitative variable in the dataset specified by the user instead of "Year".

We also added an explicit example in the manuscript to clarify that plotting parameters (like the horizontal axis label) can be tailored from their default value of the inputted to more specific values. This can be seen in Figure 5 (has the default horizontal axis value of the column name of the quantitative variable in the dataset, "devYear") and Figure 6 (has been tailored to have a horizontal axis value of "Development Year"). We accompanied this example with a brief paragraph as follows:

“Notice again from Figure 5 that the default horizontal axis label for the plotPath() method has a value of “devYear”. We wanted to change the default value of the horizontal axis label to “Development Year”. We did this in the code above for Figure 6 by appending the ggplot2::xlab() function.”

3. On page 10, last paragraph, parameter binVector is explained. I think I understand the explanation, but would suggest to simply bin, since a simple number would suffice for this purpose.

In the package, we changed the syntax by renaming the “binVector” parameter to simply “bin”. In the manuscript, we also changed all references to this parameter as “bin”.