We wish to thank all referees for their constructive comments and helpful suggestions. Almost all the comments and suggestions have been incorporated into the revised manuscript, which we think greatly improve the quality the paper. We have provided a response to each of the comments below in **blue and bold** text:

One of the major changes is that we have replaced the original *jmc()* and *jmo()* functions by *jmc\_long()* and *jmo\_long()* functions, i.e. the original *jmc\_long()* and *jmo\_long()* functions which is more user-friendly, are now used as the core functions of the package and the previous *jmc()* and *jmo()* functions are renamed as jmc\_0() and *jmo\_0()* in the revision.

**Reviewer #1:**

Q1: I wasn’t able to follow the example for jmc() exactly as outlined in the paper. The article has the following line:

jmcfit = jmc(p=8,yfile,cfile,mfile,point=20,do.trace = F)

But when I try to run this I get the following error message:

Error in jmc(p = 8, yfile, cfile, mfile, point = 20, do.trace = F) :

Possibe wrong dimension of fixed effects in Y!

This is a helpful error message that made me know that p was not specified properly. However, the paper states that “p the dimension of fixed effects (include intercept) in yfile” but the help file for jmc states that “The dimension of fixed effects (not including intercept) in yfile.” So there is a discrepancy between the article and the help file about whether p should include the intercept or not. I was able to run jmc using the specified line of code but with p=7. It took 272.54 seconds to run on my local Windows machine. I haven’t used other R programs for joint modeling so I do not know how this compares in terms of computing time.

**A1: Thanks for the errors you pointed out. There was a discrepancy between the article and the help file about whether p should include the intercept or not. We expected that p should include the intercept in both sides but the help file for jmc() in the previous version (now the *jmc\_0()* function) stated the dimension of fixed effects p inconsistently. Therefore, the correct statement in the help file should be “p the dimension of fixed effects (include intercept) in yfile”. It has been corrected in the help file for jmc\_0().**

**Also, we expected all 7 fixed effects covariates and the intercept term to be included in “yfile” so p should be 8. However, this dataset was problematic as it missed out one covariate (time.1) mistakenly and thus only 6 fixed effect covariates and the intercept term left. In this case, specifying p as 8 would give out an error message due to the inconsistency between input argument of p and data structure of “yfile”. To address this issue, we modified the dataset by adding back the covariate time.1.**

**Compared to some other R programs for joint modeling, our current algorithm can sometime be slower because our correlated random effects model typically involves more random effects than a shared random effects model and our competing risks sub-model assumes a nonparametric baseline hazard.**

Q2: I do not get the same results as those presented in the paper when I run the jmc example, likely due to the abovementioned issue with p. Comparing the two sets of results, I see that the ytime file does not contain a variable named time.1, i.e. it does not include time as a fixed effect. This will also account for why specifying p=8 did not work. These discrepancies need to be resolved.

**A2: Thanks for the discrepancies you pointed out. The reason why you couldn’t get the same results is that one covariate (time.1) was missed out mistakenly. We added the covariate in “yfile” and now the data structure becomes consistent with the specification of p=8. This modification resolves the discrepancy.**

Q3: The help file for jmc is also not clear that in fact the arguments to yfile, cfile, and mfile should be file paths, not actual dataframes/matrices or vectors. As worded I believed I needed to actually use read.table() to load the files prior to use of jmc, which is not the case.

**A3: Thanks for the issue you pointed out. It is not clear that whether the arguments to yfile, cfile, and mfile should be file paths or actual dataframes. They were designed to adapt to both file paths and actual dataframes and they were regulated by another argument “type\_file”, where type\_file = TRUE (default) is to call file paths and type\_file = FALSE is to call dataframes. To clarify the use of jmc\_0, we incorporated extra example codes into the help file of jmc\_0, where both ways of calling data were considered.**

Q4: I was able to run the jmo example as presented in the article with identical results. It took 2291.03 seconds to run on my Windows machine, which is quite slow.

**A4: Thanks for the testing information you provided. The reason why the program takes a long time is that our joint model is 1) a correlated random effects model and 2) based on semi-parametric framework, which a flexible separate random effects structure for the longitudinal sub-model and the survival sub-model and nonparametric baseline hazards for the competing risks sub-models. The estimation methods for such a flexible model involve high computational complexity, such as multi-dimensional numerical integration in the E step and nonparametric estimation of the cumulative baseline hazard functions in the M step. However, some of the issues can be addressed by linearizing the computation and we will work on the algorithmic improvement in our future work.**

Q5: The code example for adding a categorical variable in the article needs some revision:

- The line mread <- read.table(file = "fvc621\_m.txt", header = T) should actually be mread <- read.table(file = system.file("extdata", "fvc621\_m.txt", package = "JMcmprsk"), header = T)

- You use the function left\_join from the dplyr package without ever showing loading that library. It would be best to change the line yread <- left\_join(yread, cate\_var, by = "rowId") to yread <- dplyr::left\_join(yread, cate\_var, by = "rowId") to show this explicitly, or use merge from base R

- I actually don’t understand the point of creating the object mread. Can you explain?

- The object mread has 139 rows but yread has 140 unique individuals and cread has 140 rows. When I run the line cate\_var <- data.frame(rowId, sex, race), I get the following error:

Error in data.frame(rowId, sex, race) :

arguments imply differing number of rows: 140, 139

- I would suggest proceeding in this way instead, as the line that starts mread is superfluous and we can simply use the number of rows in cread (140, the correct number):

set.seed(100)

sex <- sample(c("Feamle", "Male"), nrow(cread), replace = T)

race <- sample(c("White", "Black", "Asian", "Hispanic"), nrow(cread), replace = T)

rowId <- c(1:nrow(cread))

cate\_var <- data.frame(rowId, sex, race)

yread <- dplyr::left\_join(yread, cate\_var, by = "rowId")

**A5: Thanks for all the issues you pointed out. Here we listed out the questions you raised and answered them point by point as shown below.**

**(i). We realized that the line about mread is superfluous because we just wanted to obtain the number of subjects, which can be done using cread. As you suggested, we no longer keep the mread and obtain the number of subjects using cread.**

**(ii). Calling left\_join() requires dplyr package. We didn’t load this package when we used the function. Therefore, we replaced left\_join() with dplyr::left\_join() as you suggested.**

**(iii). This example includes 140 unique individuals. The reason why the object mread had 139 rows since the command line mread <- read.table(file = "fvc621\_m.txt", header = TRUE) took the first row as the header, which was unnecessary. However, this issue is unrelated now since we decided not to consider mread in our example.**

**(iiii). To make the example for adding categorical variables more illustrative, the example codes have been modified as what you suggested, and the model results have been updated accordingly.**

Q6: I got an error when I tried to run the line of code with the new categorical variables added: Error in writeLines("The reference group for", j, "is", unique(long\_data[, : invalid 'useBytes' argument

**A6: Thank you for pointing out the error. The error occurred because the previous jmc\_long() (now jmc() function) failed to handle the categorical variables with different vector types in R. We modified the codes in the current jmc() to adapt to categorical variables with all vector types. The error has been fixed.**

Q7: When you create categorical variables for the jmo\_long example, you again have a usage of left\_join from the dplyr package, and should include dplyr::left\_join instead to be explicit. Also, I got an error when I tried to run the jmo\_long line of code with the categorical variables added to the data:

Error in if (as.numeric(long\_data[, which(names(long\_data) %in% j)])[i] > :

missing value where TRUE/FALSE needed

In addition: Warning message:

In jmo\_long(yread, cread, out = "Y", FE = c("group", "time3", "time6", :

NAs introduced by coercion

**A7: Thanks for pointing out the issues.**

**(i). Similar to the aforementioned issue in A6, calling left\_join() requires dplyr package. We didn’t load this package when we used the function. The example codes have been shown to include dplyr::left\_join() to be explicit.**

**(ii). Similar to the aforementioned issue in A5, we modified the codes in the current jmo() function to adapt to categorical variables with all vector types. The error has been fixed.**

Q8: At the bottom of page 9 in the simulated data example code you have hardcoded file paths that will not work for the reader of this article trying to run the code. You should note in your article that these should be replaced with the desired destination path and file name for the various simulated data files. I had to read the help file for SimDataC to understand what those arguments were as it was not clear from the article. Similar comment for the example of SimDataO.

**A8: Thanks for pointing out the issues. We have removed the destination path for filenames in both examples for simplicity. Hence, users can replicate our results regardless their current platform or directories. All generated data will be placed in the user’s current directory.**

Q9: The “toy example on simulated data” on the help page for jmc() results in all NAs when I tried to run it. It’s important for simple examples like this using simulated data to run properly so you can test that the correct results are being produced by the functions over time. I don’t see any explicit unit testing incorporated in this package, so the examples in the help files are the closest thing.

**A9: Thanks for pointing out the issues. Current estimation for joint model requires numerical integration. Here we adopted Gauss-Hermite quadrature method as a tool to obtain the parameter estimates by doing numerical integration. In the input arguments, “point” denotes the number of quadrature points to be used for a model. More points will lead to higher accuracy of the estimates. However, if the number of points is not sufficient for an estimation, a model fit will fail to converge and not be able to give out parameter estimates, e.g., result in all NAs eventually. On the help page for jmc\_0(), we didn’t provide enough number of quadrature points, which wasn’t an illustrative example. To testify our package, we fixed this issue by providing enough points (20) to produce valid results.**

Q10: It would be helpful to include information for users about where the software is being developed (GitHub, etc) and how to submit bug/issue reports.

**A10: We have added the following information,**

**URL: https://github.com/whcsu/JMcmprsk**

**BugReports: https://github.com/whcsu/JMcmprsk/issues**

**i.e. where the software is being developed and where the bug can be reported in the package description file.**

Q11: You refer to the package Rcpp twice in your manuscript without including the appropriate citations. See citation(“Rcpp”) in R for details on the associated package citations.

**A11: We have added the corresponding citation for Rcpp package in the reference.**

Q12: The example in the jmc\_long() help file does not run as written. See, for example, the third line of the example where you refer to an explicit file path rather than referring to the cfile filepath that was created on the previous line. Ensure all examples can run when copy/pasted into R in a new R session.

**A12: Thanks for pointing out the issues. The command lines of the example in jmc\_long() (now jmc()) help file you mentioned were incoherent, which may cause some confusions to users. To make the example more illustrative, we modified the example codes by reading the datasets from the file paths created on the previous lines and called the datasets to jmc\_long. Afterwards, we double checked the codes, and they are runnable.**

Q13: The real data example in the jmo help file produced all NAs when I ran it locally

**A13: Thanks for pointing out the issue. Similar to the issues in Q9, the current estimation for joint model requires numerical integration. Here we adopted Gauss-Hermite quadrature method as a tool to obtain the parameter estimates by doing numerical integration. The reason of all NAs produced is insufficient number of quadrature points. To address this issue, we provided enough points (20) to produce valid results for the real data example (not run) in the jmo help file.**

Q14: 1. The example code in jmo\_long doesn’t run. I get the following messages:

Error in if (as.numeric(long\_data[, which(names(long\_data) %in% j)])[i] > :

missing value where TRUE/FALSE needed

In addition: Warning message:

In jmo\_long(yread, cread, out = "Y", FE = c("group", "time3", "time6", :

NAs introduced by coercion

**A14: Thanks for pointing out the error. Similar to the issues in A5, the reason of the error was that previous jmo\_long() (now the jmo()function) failed to handle the categorical variables with different vector types in R. We modified the codes in jmo\_long() (now the jmo() function) to adapt to categorical variables with all vector types. The error has been fixed.**

Q17: The code in this package would benefit in readability and best practices from adherence to the tidyverse style guide https://style.tidyverse.org/index.html. The styler package https://github.com/r-lib/styler could be usefully applied to this end.

**A17: We have used styler to format the R files in the current package. The most recent version at CRAN reflects this change.**

**Reviewer #2:**

Q1: The first example, as written, does not currently work (p=6 or 7, not 8, with the data that is provided). In the help file for jmc it says that p=the dimension of fixed effects (not including intercept) however the manuscript states that p is the dimension of fixed effects (including intercept) in the yfile.

**A1: Thanks for pointing out the errors. There was a discrepancy between the article and the help file about whether p should include the intercept or not. We expected that p should include the intercept in both sides but the help file for jmc() (now jmc\_0()) stated the dimension of fixed effects p inconsistently. Therefore, the correct statement in the help file should be “p the dimension of fixed effects (include intercept) in yfile”. It has been corrected in the help file for jmc\_0().**

**Also, we expected all 7 fixed effects covariates and the intercept term to be included in “yfile” so p should be 8. However, this dataset was problematic as it missed out one covariate (time.1) mistakenly and thus only 6 fixed effect covariates and the intercept term left. In this case, specifying p as 8 would give out an error message due to the inconsistency between input argument of p and data structure of “yfile”. To address this issue, we modified the dataset by adding back the covariate time.1.**

Q2: There are some minor grammatical issues that need to be fixed (I've highlighted some in the pdf). Primarily missing are definite/indefinite articles (the, a, an).

**A2: Thanks for pointing out the grammatical issues. We didn’t examine these issues carefully before. We fixed all highlighted parts according to your comments.**

Q3: It would be helpful to provide some information on how long it takes to run these functions (is there a relationship to overall N or the # events?). The jmo function, in particular, takes a long time to run (p=9, over 40 minutes). The jmc function using p=6 took over an hour to run vs. ~6 minutes with p=7, but both run.

**A3: Thanks for the comments! We have tested the time complexity of our models on a variety of scenarios for overall N and # of events, the corresponding run times of these scenarios are summary in the following tables. We have also added a new subsection on “computation complexity” in the revision to answer the reviewer’s concern.**

**According to Tables 1 and 2 below, we can see that the runtime is clearly positively associated with the sample size and it increases much faster than linear order, given a fixed # of events.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **N** | **Event rate** | **42.6% (37.6% + 5%)** | **60.8% (48.6% + 12.2%)** | **77.6% (57.2% + 20.4%)** | **94% (66.2% + 27.8%)** |
| **500** | | **74** | **72** | **67** | **59** |
| **1000** | | **181** | **188** | **162** | **195** |
| **1500** | | **356** | **343** | **228** | **291** |
| **2000** | | **502** | **570** | **500** | **311** |
| **2500** | | **856** | **928** | **841** | **968** |
| **3000** | | **1022** | **1392** | **1306** | **1356** |
| **3500** | | **1154** | **2028** | **1848** | **1408** |
| **4000** | | **1722** | **2731** | **2530** | **2560** |
| **4500** | | **2482** | **3642** | **3289** | **3471** |
| **5000** | | **3552** | **4759** | **4557** | **4572** |

**Table 1. Run time (s) under different sample sizes and # of events (risk1 + risk2) for jmo(). Each simulation takes 4 covariates and 10 quadrature points.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **N** | **Event rate** | **58.4% (35% + 23.4%)** | **68.8% (41.2% + 27.6%)** | **79.6% (45% + 34.6%)** | **89.6% (51.4% + 38.2%)** |
| **500** | | **35** | **27** | **24** | **24** |
| **1000** | | **217** | **88** | **87** | **98** |
| **1500** | | **217** | **339** | **481** | **149** |
| **2000** | | **370** | **371** | **1457** | **485** |
| **2500** | | **926** | **471** | **1326** | **779** |
| **3000** | | **1727** | **1035** | **1445** | **989** |
| **3500** | | **1108** | **1826** | **2338** | **1582** |
| **4000** | | **2181** | **2095** | **3216** | **2075** |
| **4500** | | **3678** | **2972** | **3770** | **3078** |
| **5000** | | **4810** | **4347** | **5655** | **4624** |

**Table 2. Run time (s) under different sample sizes and # of events (risk1 + risk2) for jmc(). Each simulation takes 4 covariates and 6 quadrature points.**

Q4: no summary information is provided about the call or the datasets - such as number of observations - so it is harder to identify the data source if the results are saved and viewed later. At a minimum, I'd recommend storing the function call and the paths to the data in the results object.

**A4: Thanks for your comments. It is confusing for users to identify what data they used and functions they call when the results are saved. To clarify the summary information of the result object, we incorporated the following items referencing other R programs for joint models: function call, paths to the data, number of observations, number of groups, proportion of competing risks, method for numerical integration, model summary, and loglikelihood.**

Q5: no ID is required (instead, you are counting the number of rows for each subject and listing that in the mfile), so there doesn't appear to be any mapping between "ninds\_nrank\_c" (N=586) and "ninds\_nrank\_y" (N=1906) - there are no checks that these are even the same subjects. Instead of using the mfile, I think it would be safer to include an ID column in the yfile and in the cfile.

**Q5: At the suggestions of the reviewer, the current version of jmo() and jmc() functions does not require mfile anymore. In this case, we added some R commands to check if there is a correct mapping between “ninds\_nrank\_c” and “ninds\_nrank\_y” with ID columns in both files before running a model in jmo(). Moreover, we also implemented the same procedure in jmc().**

Q6: Is there a helper function to extract the estimates, std.err, CI and p-values for both the longitudinal and survival portions of the results? For instance, if I wanted to present these results in a report then I would want to change the digits that are presented for the p-value, change the estimates to HR for the survival fixed effects, and also change the digits. The print method, as implemented, doesn't allow the user to specify digits. For instance, if you look at print.coxph and print.lm, both allow the user to modify digits (common for other model fits as well).

> args(survival:::print.coxph)

function (x, digits = max(1L, getOption("digits") - 3L), signif.stars = FALSE, ...)

> args(stats:::print.lm)

function (x, digits = max(3L, getOption("digits") - 3L), ...)

**Q6: Thanks for pointing out the issues. We listed the questions you raised and provided the corresponding solution to those as shown below:**

**i). The result object had the parameter estimates and standard errors for both the longitudinal and survival portions, but we didn’t provide a helper function to extract those values initially, which may not be user-friendly. To address this issue, we developed a R function summary() to let users obtain the longitudinal / survival portion of the results based on “coeff” argument they specify. This function enables us to show the parameter estimates, standard errors, 95% CIs and p-values for both portions. Especially for the survival portion, we also offered the HR for survival fixed effects. Additionally, we added another argument “digits”, which allows users to change the digits.**

**ii). Our print method doesn’t allow users to change the digits, compared to other R programs for extracting model results. Thus, similar to the digits specification in summary(), we added the argument “digits” into our print method to make it comparable to other existing print methods.**

Q7: In the first example, it is unclear when you would ever want to test that all of the coefficients are = 0 for a particular model (implementation of the anova method). I would have instead expected the anova() to act as it does with other models in R, collapsing factors for multiple df tests. Though the way the data has been implemented, this isn't really possible.

**Q7: Thanks for pointing out the issues. The first example wasn’t illustrative since we are not sure when we should test all the coefficients = 0. We didn’t consider the hypothesis testing of the coefficients very carefully and this example may even bring some confusions to users. Instead, we replaced our anova() with linearTest(), in which users are allowed to specify and test a linear hypothesis by collapsing factors for multiple df tests.**

Q8: How are missing values handled? Is it required that the model for the longitudinal data is the same as it is for the survival data?

**Q8: Thanks for raising the questions.**

**Our current version of package requires both longitudinal and survival data to be complete. Otherwise, the functions will stop with an error message as the package detects missing values.**

Q9: It is unclear why the user must temporarily save the data to external files - it shouldn't be necessary. It would be less awkward for the user to have a wrapper function that does some basic data checks and creates the necessary data format, then have an internal function that interfaces with the C++ code. That said, the code does work, especially using the data preparation functions that are provided.

**Q9: Thanks for the suggestions of the reviewer, the current version of jmo() and jmc() functions does not require users to save data temporarily to external files.**

Q10:Just as a thought (doesn't need to implement) - the function call could be something like the following. This would then make it easier for the user to add in splines or other variable transformations, and it would allow you to recognize when one of the covariates is a multi-level factor that should be collapsed in the anova() function.

jmc(Surv(time,event)~(FVC0 + FIB0)\*CYC, id1=ID, data1=c.dat,

FVC ~ (time\_RE + FVC0 + FIB0)\*CYC + (time\_RE | ID), data2=y.dat)

**Q10: Thanks for the suggestions. The function call like the above is very straightforward and easy to specify, as shown in other R packages for joint models. We appreciated the example you gave here and thus it will be one of our future work to improve the interface of our package from user’s perspective.**