Cover Letter

Dear Editor:

I would like to resubmit our manuscript (RJournal 2021-188) entitled "SurvMetrics: An R package for Predictive Evaluation Metrics in Survival Analysis" for possible publication in your journal.

We sincerely appreciate you for giving us the opportunity to improve our work. Following the reviewers' comments and suggestions, we have revised the manuscript substantially. Major modifications include: we have added two input formats, standard models and non-standard models, to the example code. We have also responded point-by-point to each comment made by the reviewers in the attached files.

Please note that the research work described has not been submitted elsewhere for publication, in whole or in part, and all the authors listed have approved the manuscript that is enclosed. The authors thank the editor, the associate editor, and reviewers for their constructive comments, which have led to a dramatic improvement in the earlier version of this article. I hope you find the paper as groundbreaking as we do and I look forward to hearing your decision.

Corresponding Author: Hong Wang
Ph.D., Associate Professor
School of Mathematics & Statistics
Central South University
Hunan, China
Email: wh@csu.edu.cn

Reviewer 3

Thanks to the authors for addressing all of my points. Only one issue I noticed in the revision.

Comment :

The code on pages 7-8 of the manuscript doesn't show the easier-to-use implementation that was developed in the revision, e.g. the C index and Brier score can be computed more directly from the fitted model as:

```
Cindex(fit_cox, test_data)
Brier(fit_cox, test_data, distime[med_index])
```

I'd show both implementations, and explain that the longer version is still useful to deal with predicted survival probabilities from non-standard models.

Reply:

We sincerely appreciate your thoughtful comments. Following your suggestion, we have added two input formats, standard models and non-standard models, to the code on pages 7-8 of the manuscript. The specific contents of the manuscript are as follows.

In the following, we first cut the kidney dataset into a training set and a testing set. Then, two popular models, namely, Cox model and random survival forest model are constructed based on the training set to show how different functions are used in SurvMetrics. Finally, we will show how to evaluate the predictive performance of these two models on the testing set using the proposed SurvMetrics R package. In the latest version of SurvMetrics, user has the choice to input only the standard survival models and testing sets. Meanwhile, it is still useful to deal with predicted survival probabilities from non-standard models. The following example will contain these two input forms.

The corresponding R code is provided here:

```
#1. data preparation
library(survival)        # to fit a Cox model
library(randomForestSRC) # to fit an RSF model
library(SurvMetrics) # to get all the metrics
library(pec)         # to make predictions based on Cox model
set.seed(1)
mydata <- kidney[, -1]
train_index <- sample(1:nrow(mydata), 0.7 * nrow(mydata))
```

```
train_data <- mydata[train_index, ]
test_data <- mydata[-train_index, ]

#2. fit the RSF model and Cox model to predict the testing set
#2.1 RSF model
fit_rsf <- rfsrc(Surv(time,status)~., data = train_data)  #fit the RSF model
distime <- fit_rsf$time.interest  #get the survival time of events
med_index <- median(1:length(distime))  #the index of median survival time of events
mat_rsf <- predict(fit_rsf, test_data)$survival  #get the survival probability matrix
vec_rsf <- mat_rsf[ ,med_index]  #median survival probability of all samples

#2.2 Cox model
fit_cox <- coxph(Surv(time,status)~., data = train_data, x = TRUE)
#fit the Cox model
mat_cox <- predictSurvProb(fit_cox, test_data, distime)
#get the survival probability matrix
vec_cox <- mat_cox[ ,med_index]

#3. get all the metrics by SurvMetrics
#3.1 CI BS IBS IAE ISE based on RSF model: standard model input methods
Cindex_rsf_sd <- Cindex(fit_rsf, test_data)
BS_rsf_sd <- Brier(fit_rsf, test_data, distime[med_index])
IBS_rsf_sd <- IBS(fit_rsf, test_data)
IAE_rsf_sd <- IAEISE(fit_rsf, test_data)[1]
ISE_rsf_sd <- IAEISE(fit_rsf, test_data)[2]

#CI BS IBS IAE ISE based on Cox model: standard model input methods
Cindex_cox_sd <- Cindex(fit_cox, test_data)
BS_cox_sd <- Brier(fit_cox, test_data, distime[med_index])
IBS_cox_sd <- IBS(fit_cox, test_data)
IAE_cox_sd <- IAEISE(fit_cox, test_data)[1]
ISE_cox_sd <- IAEISE(fit_cox, test_data)[2]

#3.2 CI BS IBS IAE ISE based on RSF model: Non-standard model input methods
times <- test_data$time
status <- test_data$status
Cindex_rsf <- Cindex(Surv(times, status), vec_rsf)
BS_rsf <- Brier(Surv(times, status), vec_rsf, distime[med_index])
IBS_rsf <- IBS(Surv(times, status), mat_rsf, distime)
# distime can be replaced by range(distime)
IAE_rsf <- IAEISE(Surv(times, status), mat_rsf, distime)[1]
```

```
ISE_rsf <- IAEISE(Surv(times, status), mat_rsf, distime)[2]

#CI BS IBS IAE ISE based on Cox model: Non-standard model input methods
Cindex_cox <- Cindex(Surv(times, status), vec_cox)
BS_cox <- Brier(Surv(times, status), vec_cox, distime[med_index])
IBS_cox <- IBS(Surv(times, status), mat_cox, distime)
IAE_cox <- IAEISE(Surv(times, status), mat_cox, distime)[1]
ISE_cox <- IAEISE(Surv(times, status), mat_cox, distime)[2]
```