# Responses to the comments of review-1

We thank the reviewer for her/his careful reading of our manuscript and her/his many insightful comments and suggestions. Below we respond to the comments with Reviewer's comments in bold.

**I felt like I had to go read the JSS paper to understand the background section - it would have been nice for them to actually include table 1 from that paper, which actually lists the algorithm, instead of just the formulas. Definitely include it in the revision, and beef up the motivation.**
Answer: We added a table (Table 1) with the pseudo-code.

**The simulations and examples all ran fine / replicated on my computer. They are all small synthetic data sets, the largest is only 100x1000, so the timings are quite noisy. I'd recommend using either trimmed means / sds for benchmarks. A 30% improvement is nice but doesn't seem too impressive in the context of milliseconds. They should show larger simulations where it makes a bigger difference.**
Answer: We increased the dimension of the simulated data so that the largest is 200x7000. Additionally, we have used the 5% trimmed mean and sd.

**The GRCV example makes sense to me, and I can understand how it's useful for hyperparameter tuning. When I compared the actual code in the package to the written description, it wasn't clear to me if they meant to do the cross-validation split inside the iteration loop or not.**
Answer: In the second stage of the GRCV estimation, two criteria (BIC and AIC) can be used. However, in Pazira et al. (2018) we recommended using the AIC because in the third stage the goal is to have an accurate prediction.

**I'll be reviewing the code mirrored at *https://github.com/cran/dglars* - it seems like the authors don't have a public repository of their own. The article you linked would call that an auto-reject but I feel that is pretty harsh.**
Answer: We have a public repository on:
`https://cran.r-project.org/web/packages/dglars/index.html`.

**'*!inherits*' instead of '*class(object) !=*'**
Answer: `inherits` function is used to check the class of an `R` object.

**I'd recommend a helper function instead of the 'for(i in 1:2)' flow construct.**
Answer: Done

**Maybe use message() instead of cat - will go to stderr instead.**
Answer: Done

**They could use match.fun for the type parameter, in case the user has a custom goodness of fit function.**
Answer: Done

**Allow the phihat function to accept multiple types at once; return a dataframe if more than one specified.**
Answer: Done

**In grcv(), when I look at the intermediate phi - it seems like it is mostly duplicated when using the Gamma example in ?dglars - is the random split supposed to happen within the iteration loop? Seems like a bug?**
Answer: Done

**Survival model / cox model - the author has another paper on this, so perhaps it will be in the next version.**
Answer: We have a public repository on: `https://github.com/HassanPazira/DgCox` for the survival model called "dgcox".

**Overall, I can see why this package exists and what its contributions are. Their paper hasn't convinced me to switch to their package from glmnet, though.**
Answer: In our another work, Pazira et al. (2018), we compared our method with `glmpath`, which uses the PC algorithm, because `glmnet` implements the

cyclical coordinate descent algorithm.

**I may try implementing grcv() for glmnet fits, I don't think it's specific to dglars.**
Answer: The GRCV can be implemented for other methods.

We have found the rest of the comments (the following comments) really interesting and can improve the speed/efficiency of the package. But since fixing/considering them take much time, we need more time to implement these comments and update our codes. However we started to edit our codes and we are still working on them.

**Support arbitrary family / link functions. This would probably require a lot of refactoring.**

**The cross validation function is implemented in fortran, runs serially. Fixing this would improve performance much more than IPCA.**

**They could also save out the RNG state as an attribute, to make replication easier.**

**Sparse matrix support & Support parallelizing cross validation**

**Each family / link function combination is specially implemented as a FORTRAN subroutine. Each of those has a large amount of duplicated / redundant code. This is a brittle architecture.**

**The Fortran is not great, but average for academic code. I'd suggest they match the style of netlib, and add specific comments for which chunks of code go with which formulas in the JSS paper. It seems like only helper fortran functions are documented, and not the actual implementations of dglars.**

**The *.Call* api is preferred over *.Fortran* - it used to make fewer intermediate copies of the data, although that may have been fixed since. It also allows a more flexible return types eg named lists.**

**Refactor summary method to have own S3 class / print method**

**A helper method for control settings, with it's own manual page, like lme4 does**
Answer: We still working on the above comments.