Response to AE and Reviewer – FarmTest: An R Package for Factor-Adjusted Robust Multiple Testing

Reviewer Comments: The paper proposed an R package named FarmTest to implement multiple test when there exist possible strong correlation among tests. A multi-factor model is used to capture the dependence among a large pool of variables. Both observed factors and unobserved factors are analyzed. Overall, the paper and package are well written and the method is sensible and novel.

RESPONSE: We are grateful for your positive and encouraging comments. The paper and the package have been revised accordingly to address all the questions. We hope it will be received favorably by the reviewer, and we look forward to your decision. Please see our detailed response given below.

General comments and overview:

1. *When there exists a latent factor structure, an essential step is to select the number of factors K. The authors used a eigenvalue ratio criterion to estimate K. In practice, the estimated K is usually larger than 0, how to estimate K when the true number of factors is* 0*?*

   RESPONSE: Thank you for pointing this out. When there is indeed a latent factor structure, the eigenvalue ratio criterion is one of the most effective ways to estimate the number of factor, and produces a $\hat{K} \geq 1$. If we have prior knowledge that the variables of interest are independent or weakly correlated, we can manually set the argument `KX = 0` in the commend: `farm.test(X, KX = 0)`. Alternatively, Bai and Ng (2001) proposed a BIC-type estimator of K, which is also consistent when $K = 0$. This approach is two-step: first, fit factors via minimizing sum of squared residuals for different values of k, and then minimize a penalized MSE to find the best $0 \leq k \leq k_{\max}$. This approach is computationally

much more expensive than the eigenvalue ratio criterion, and hence we use the latter in our package. To also cover the null case of $K = 0$, we can slightly modify the current method as follows: set $\hat{K} = 0$ if the explained relative variance of the (estimated) leading eigenvalue is less than a prespecified threshold, say 0.1.

In general, if we don't know whether the true K is zero or not, we may consider testing the hypotheses $H_0 : K = 0$ versus $H_1 : K \geq 1$, which is a problem of independent interest and challenge. An important work on this topic is Onatski (2009) which is tailored to a right-sided alternative. A more general testing method that is also applicable to left-sided and two-sided alternatives can be found in Kong (2020).

2. *The package implements a series of adaptive Huber methods for estimation, for a real data sets, can we have a date driven procedure to determine which one should be used? Which one is the default method? Or we just use the simply least square type method?*

   RESPONSE: The current package implements Huber-type methods (with a tuning-free scheme) for three problems: mean estimation, covariance estimation and linear regression. To achieve robustness against heavy-tailed and/or asymmetric errors, these tuning-free Huber-type methods are recommended and set as default. Numerical advantages over least squares type method can be found in recent studies (Ke et al., 2019; Wang et al., 2020).The adaptive Huber method performs as good as the least squares counterpart when data are from a normal model. As the level of skewness or heavy-tailedness increases, the advantage of adaptive Huber method becomes more evident.

3. *Do we need to normalize the data or the data has been normalized automatically in the package before analysis?*

   RESPONSE: The data are normalized automatically in the package. The proposed multiple testing problem can be decomposed into several sub-problems, and one of them is a Huber regression. It turns out that normalizing the design matrix and centering the response before applying the gradient descent algorithm significantly improves the computational speed over the iterative reweighted least squares method, especially when K is large.

4. *The procedure involves eigenvalue decomposition procedure for a covariance matrix of dimension p, how to calculate it when p is large? How is the default method for this eigenvalue decomposition procedure?*

RESPONSE: Eigenvalue decomposition is involved for models with latent factors. Since the estimated covariance matrix is guaranteed to be symmetric, we use the function `eig_sym` in the C++ Armadillo library (Sanderson and Curtin, 2016). The default method for eigenvalue decomposition is a divide-and-conquer approach originally proposed in Cuppen (1981). Briefly speaking, the problem is divided into two problems of roughly half the size, each of which is solved recursively, and the eigenvalues of the original problem are computed from the results of these smaller problems. Divide-and-conquer is also the method when p is large. Since this decomposition is operated by an existing function, we omit the details in the paper.

Comments for the article:

The examples are helpful and clear. I have only one comment. The most commonly used methods for FDR control are those developed by Benjamini and Hochberg (1995) and Storey (2002). I understand that the approach proposed in this package is just modified follow the direction of Storey (2002). Maybe it is more proper to add the method of Benjamini and Hochberg (1995) also in this package for comparison.

RESPONSE: Thank you for the advice. The p-values that correspond to the proposed factor-adjust robust test statistics are included in our output; see Table 1 in the paper. In principle, one can apply any existing FDR/FDP control method to these p-values, thereby obtaining a list of rejected hypotheses at a predetermined level α. The classical FDR control method by Benjamini and Hochberg (1995) can be implemented by the function `p.adjust` in the base package `stats`. So for comparison, the users may run one more step to get the results from the BH procedure:

```
output = farm.test(X)
p.adjust(output$pValues, method = "BH")
```

Comments for the package:

The package is well written and have no other comments.

RESPONSE: Thank you for your encouragement. We have an update for the `huber.reg` function in this package, with an option `method = c("standard", "adaptive")` added to distinguish the two different ways for choosing the robustification parameter τ. The details of these two methods can be found in Sections 3.1.1 and 3.1.2 in Wang et al. (2020). This version `FarmTest 2.2.0` has been available on CRAN now, and we will keep it updated with any feedback we receive.

References

Bai, J. and Ng, S. (2001). Determining the number of factors in approximate factor models. *Econometrica* **70** 191–221.

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological)* **57**(1) 289–300.

Cuppen, J. J. M. (1981). A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numerische Mathematik* **36** 177–195.

Ke, Y., Minsker, S., Ren, Z., Sun, Q., and Zhou, W.-X. (2019). User-friendly covariance estimation for heavy-tailed distributions. *Statistical Science* **34**(3) 454–471.

Kong, X. (2020). A random-perturbation-based rank estimator of the number of factors. *Biometrika* **107** 505–511.

Onatski, A. (2009). Testing hypotheses about the number of factors in large factor models. *Econometrica* **77** 1447–1479.

Sanderson, C. and Curtin, R. (2016). Armadillo: A template-based C++ library for linear algebra. *Journal of Open Source Software* **1**(2) 26.

Storey, J. (2002). A direct approach to false discovery rates. *Journal of the Royal Statistical Society: Series B (Methodological)* **64**(3) 479–498.

Wang, L., Zheng, Z., Zhou, W., and Zhou, W.-X. (2020). A new principle for tuning-free Huber regression. *Statistica Sinica*, to appear. http://www.stat.sinica.edu.tw/statistica/ss_newpaper/SS-2019-0045_na.pdf