Responses to reviewer comments

A big thank you to the reviewers for taking the time to study this manuscript and contribute their feedback. The reviews demonstrated great technical knowledge of the R software as well insight in the development process. Most suggestions were very valuable and have been incorporated in the paper. Below a point-by-point response to the reviewers regarding the specific comments.

Reviewer 1

[Comment] *Article: I think nothing essential is missing from the article. If I were to change anything substantive, I might ask that the author provide more details about the precise disk and memory costs of switching R to the dependency model used by npm for storing dependencies of a package within the package. Is this done using symlinks or by literal copying the same files over and over again?*

[Response] In the case of NPM, dependency packages are indeed actually installed in the private library of a package. The goal is to create a self-contained directory that has its dependencies included, and is no way affected by changes of other packages. This does not always come down to copying the same files over and over again, as different packages will often require different versions of the dependency package. But in the case where multiple packages depend on exactly the same version of a dependency, their private libraries contain duplicates, indeed. In the case of R, we could also imagine an alternative layout in which there is still a single global library, containing multiple versions of each package. The package manager would then be responsible for automatically locating and loading the appropriate dependency versions along with a package. This layout could save some disk space by preventing duplicates.

It is hard to really predict the trade-off in memory and disk if this were to be implemented in R. Obviously, if multiple versions of a package need to be installed, this would require more disk space than if only one version of each packages needs to be available. Similarly, if two packages are loaded that depend on different versions of, say, MASS, this would require more memory than when both packages can use the same version of MASS. However, for most users the amount of memory needed to load a package in memory is minor in comparison with memory required to load and manipulate a medium sized dataset.

Reviewer 2

[Comment] Fundamental misunderstanding of versioned Depends *Surprisingly, the paper appears unaware of the current feature set of R. Dependencies are not just "by name only" (p2) but also via a minimum version. R does have a $\geq$ operator. For example, lme4 currently has Depends: methods, R($\geq$ 2.11.1), Matrix($\geq$ 1.0-1), lattice stating a minimum version of both R and Matrix (which, via Matrix having R ($\geq$ 2.15.0) also imposes*

a tighter constrain on R). This is clearly documented in the Writing R Extension manual, Section 1.1.1 entitled "The DESCRIPTION file". Moreover, CRAN has subdirectories `src/contrib/$VERSION` *for everything from 1.4.0 (!!) to 2.15.3 (an R version not yet out) and 3.0.0. Package versions newer than the in "R-release" can often be found here.*

[Response] Reviewer gives an interesting example: the `lme4` package that is currently on CRAN depends on `Matrix(>= 1.0-1)`. Hence, this states that version `0.999999` of `lme4` will work with any future version of the `Matrix` package that will be released from now, ever to appear on CRAN, in the near or distant future. This seems like a rather dangerous assumption. Suppose we would like to install this specific version of `lme4`, say, 10 years from now, to reproduce some results. Should we blindly assume that `lme4 0.999999` will work with `Matrix 3.9999`? The answer of course is no; this version of `lme4` has only really been tested with `Matrix` versions `1.0-x`. Future versions of `Matrix` might introduce breaking changes. The main concern throughout the paper is to acknowledge this reality, and have the package manager accommodate a solution. If the `lme4` package is working now, it shouldn't suddenly break when a new version of the `Matrix` package is released. In this case, both packages are written by the same author, who can foresee problems and publish updates for both `lme4` and `Matrix` simultaneously. However this mechanism does not work for the thousands of inter-dependent packages on CRAN.

The $\geq$ operator merely provides a check to verify if the installed package is "current" enough, or whether it should be synchronized with CRAN. It does allow authors to be more specific about the dependency version, nor does it help with resolving conflicts. The problems described in the paper have little to do with installed packages being outdated. On the contrary: packages that have been developed targeting an *older* version of the dependency package are being forced to use newer versions become available. Furthermore, the $\geq$ operator is not used anywhere else in the R software to refer to a package. For example, the `install.packages` or `library` functions do not support specifying a version. For this reason it is stated that R refers to packages by name only.

Regarding the final lines: it is acknowledged in the paper that CRAN indeed has a great archive of source packages. The problem described in the paper is that the package manager does not have a system in place to properly take advantage of this archive, as everyone is expected to be using the "current" versions of packages. This is further illustrated below when talking about reproducible research.

Unaware of complete "recursive tests" at CRAN *Uwe Ligges in his keynote at useR clearly stated that he rebuilds every package up in the dependency tree. As we understand it, that is how conflicts and issues (e.g. with ggplot2 0.9.0) are found.*

[Response] This observation has been incorporated in the paper. However it does not quite provide a solution to the problem of dependency versioning for the following reasons.

First of all, only a small fraction of potential problems due to changes in dependencies

will appear during building of the package. Many problems due to behavioral changes in dependencies will go unnoticed, unless an example happens to be present that explicitly throws an error. Some packages do include a set of unit tests, that potentially could help with detecting such problems at build time. However, unit tests are the exception and not the rule; most package authors currently rely on manual testing. Moreover, CRAN maintainers have actually requested package authors to limit automatic unit testing during package building to a minimum, due to overloading the CRAN hardware. Hence, recursive rebuilding might find a problem or two, but is far from reliable.

Secondly, these recursive tests do not provide any solution to conflicts that might be found. When a problem is detected, CRAN maintainers will need to contact package authors of broken packages to manually update their packages to work with the latest version of the package they depend on. This policy results in a lot of work for everyone involved and is unsustainable as CRAN is growing. Furthermore it is simply unnecessary: If a certain version of a package has been developed for, and are tested with a certain version of dependencies, it should keep using those versions. Forcing all package authors to commit to a life long updating of their package as new versions of dependencies become available is infeasible and silly.

[Comment] Reproducible research: "sweave not building" *The same is true for R, the operating system and its version and more: "everything needs to be held constant". So this is not an issue unique to R. The solution might be the "freeze" of VMs or AMIs. R's sessionInfo() is a fair start as it gives R and package versions and those are in fact all available on CRAN in the Archive/ directories. R may well do better here than other languages.*

[Response] Reviewer makes several interesting observations here. With regard to the dependency on R or OS level functionality: indeed, in theory reproducibility can not be guaranteed unless every aspect of the machine has been "frozen". And even then there is a possibility of code interacting with external sources over the network, resulting in behavioral changes. In practice however, OS level dependencies form a much smaller and distinguishable problem from dependencies on R packages. In my personal experience, most Sweave authors are wise enough not to include hardcoded paths or shell commands in their document, especially if they intend to share the document or submit it to be considered for publication. However, almost every Sweave document I have seen uses contributed packages. It is still true that some R packages rely on operating system level functionality or libraries that could potentially alter behavior. However, the community has made great efforts to make R software portable among the various platforms. In my view, the cross-platform challenges of R packages interfacing to the OS is a distinct topic from the issue of dependency versioning, and does not undermine the importance of versioning R packages in reproducible research.

Reviewer then mentions `sessionInfo` to list the R packages in use. This is a great suggestion. Ideally all Sweave authors would include the output of their `sessionInfo` with

the Sweave document. This would provide a starting point for reproduction. However, reproduction would still not be trivial. One would then manually have to download each package from the archives as the package manager can only download the latest release of each package. Moreover, Windows and OSX users will find that for many older releases no binary packages are available, and need to rebuild these packages from source. All of this makes reproducible research impractical and not a natural part of the users workflow. A package manager that natively acknowledges dependency versioning would be a tremendous gain in this respect.

Based on suggestions by reviewer and editor, an additional paragraph has been included in the reproducible research section. This section provides an example of what would in practice be involved with reproducing a Sweave document that is 2 years old.

[Comment] Unaware of BioConductor model *It seems that the BioConductor model already uses what the author describes in "The release cycle".*

[Response] In the preparation of the original manuscript I had studied the bioconductor website and literature. However, the project and design of the software seemed very domain specific to biological computing, a field with which I am unfortunately unfamiliar. Because I have no experience with analyzing genomic data, and little understanding about the specific challenges that researchers face in this field, I did not feel comfortable commenting on this project. Based on the suggestion by reviewer I have revisited Bioconductor and consulted colleagues that make use of this software. On several places in the manuscript I have included a references to bioconductor, and how it relates to the concerns being discussed.

[Comment] Saying "things break on a regular basis, which is completely expected" is plain false *We would would expect CRAN maintainers to not be happy with this description. Things generally work with the set of packages on CRAN – that is what the CRAN volunteers accomplish with all their checks.*

[Response] Perhaps this was indeed worded too strongly; it has been modified. I intended to comment on the general case of the design of an development pool where developers independently push updates. It is wise to assume that each package update can result in depending packages breaking due to newly introduced changes. As stated above, the CRAN checks are far from a complete solution.

[Comment] Unaware of the REvolution Analytics model *They already freeze at the previous release, ie one currently cannot purchase R 2.15.* from them but only R 2.14.* for the stability imperative reasons cited here.*

[Response] It is unclear what precisely reviewer refers to with the *Revolution Analytics Model*. The Revolution Enterprise software installs the R software along with an IDE/GUI and some customized R packages. Indeed, this version of R is thereby frozen. But as far

as I can tell, the software relies on the native R package manager and the usual CRAN repositories. It does not in any way freeze R packages (apart from the handful included in the installer) or suggest any other solution for the problems described in the paper. The only relevance I see is that it is an example of software that depends on R.

[Comment] The npm model could be done in R with per-project .libPaths() settings *We consider this somewhat wasteful, and a last resort done only there can be no support from the host OS – similar to how large apps such as Chrome or RStudio come with all libraries embedded. This is also typical on Windows. But by setting a lib.loc argument to install.packages, this could already be achieved with the existing code base (apart from the no-two-versions-of-same package issue).*

[Response] Reviewer is definitely thinking in the right direction here. Maintaining separate libraries for different tasks or projects is a wise practice. However, as stated in the paper, there are currently two fundamental limitations in R that make this impractical. First, the default behavior of R discourages authors to explicitly require a specific version of a dependency about dependency versions, because in the situation of a single global library, there is only version of each package available. Consider again the example of `lme4` discussed above. How are we going to get `lme4 0.999999` working 5 years from now if we don't know which version of `Matrix` it requires? Also if we are given a Sweave document that does `library(lme4)`, it is unclear which version of the package was used.

Secondly, as mentioned, R cannot have multiple versions of the same package loaded simultaneously. This is a major limitation. If packages and code would truly be specific about the required versions of dependencies, conflicts arise rapidly. Hence, this limitation would need to be addressed before users and package authors could really move towards versioning dependencies. However, if these two features could be supported by base R, that would probably be sufficient for the community to leverage functionality provided by `libPaths()`, `sessionInfo` and others to implement a package manager that supports versioning dependencies.

[Comment] Versioned install *We recall that R tried that several years ago and gave up. But we think this can be approximated via eg* `library(MASS, lib.loc="/my/dev/tree/1.2.3-4/")`.

[Response] See previous comment. Again the fundamental limitations w.r.t. declaring versioned dependencies and loading multiple versions of a package prevent this from being a generally usable solution.

[Comment] Repackaging for distribution *It seems like the current effort by Armstrong now at* `http://debian-r.debian.net` *is near-complete. So "almost all" would be better than "some popular" when measuring CRAN package coverage. Rutter restricts himself to apprx 1/4 of CRAN for his c2d4u Ubuntu effort.*

[Response] Thanks for this pointer. However, note that unlike the packages by Eddelbuettel in the official Debian distribution, the repositories provided by Armstrong and Rutter are not frozen, but continuously updated. Hence they do not address the problem of dependency version. They are mostly there to provide precompiled binaries, in the same spirt as the binaries for Windows/OSX on CRAN.