Reviewer: - [consistency-check] The abstract and the introduction both state "more than 100 tips", but the body contains 94 numbered tips (1 through 94, the last being the `[[` versus `$` item in Section 11). The title says "90+ tips", which matches the count. Reconcile the abstract and the two introduction sentences ("a collection of more than 100 tips", "We have listed more than 100 tips") with the actual number so the claim agrees with the title and the content.

Reply: The typos is now fixed.

Reviewer: - [methods-soundness] Tip 64 states that prcomp() "performs the eigendecomposition of a matrix". prcomp() computes principal components by singular value decomposition of the (centered) data matrix, while princomp() uses the eigendecomposition of the covariance matrix. Please correct the description so the recommendation and its justification match what the functions do.

*Reply: We added this "*prcomp() "performs the eigendecomposition of a matrix". prcomp() computes principal components by singular value decomposition of the (centered) data matrix*".*

Reviewer: - [Reviewer 1] State the BLAS in use and add a reproducibility script for the reported timings. Several matrix comparisons (crossprod vs t(X) %*% Y, princomp vs prcomp, float throughput) depend on which BLAS is linked, and reference BLAS results can invert under OpenBLAS or MKL. Provide the benchmark script plus sessionInfo and the BLAS identity, or qualify these comparisons as reference-BLAS results.

Reply: We have uploaded the R code to reproduce the timing results. In the beginning of this code we have added the sessionInfo().

Reviewer: - [Reviewer 2] Add a competing-interest statement. The authors maintain Rfast, Rfast2 and estats, which the paper recommends as faster (tips 64, 72, Section 10).

Reply: We will ask the editors what action and what to write regarding this comment.

Reviewer: - [Reviewer 2] Where an authored package is recommended as faster, compare it against the strongest alternative, not base R alone. Either benchmark Rfast/estats functions

against matrixStats, collapse and data.table, or soften the superiority claims to "an alternative to base R".

Reply: We simply stated in the introduction that the comparisons are made with R's built-in functions.

Reviewer: - [Reviewer 2] Ship a runnable benchmark script that regenerates the two printed tables (tips 16 and 46) with a fixed seed, the R version, and the BLAS named, or remove the printed numbers. The uploaded file reproduces code chunks but does not run end to end.

Reply: We have updated the R code we upload now. We think we need these tables to show the time differences. We think the seed is not important, as we are measuring time. As mentioned earlier, the R code begins with the sessionInfo(), where information on the LAPACK (version 3.12.1) appears. In the tip #16 for example, we state that x contains 10,000 rows and 20 columns. So the user can generate a matrix of this size.

Reviewer: - [Reviewer 2] Correct tip 84: values below .Machine$double.eps (2.22e-16) are not rounded to zero. Underflow to zero happens near 1e-308 (and subnormals to 5e-324). The log.p advice stands, the stated threshold does not.

Reply: We changed this as the reviewer suggested.

Reviewer: - [Reviewer 3] Tip 17 option a3 divides `rowsum(x, ina, reorder = FALSE)` by `as.vector(table(ina))`, whose counts are in sorted group order, so the two are misaligned when groups do not first appear in sorted order. Fix the alignment or remove a3.

Reply: We removed the reorder=TRUE.

Reviewer: - [Reviewer 3] Add data.table coverage beyond `fread`: keyed joins, grouped aggregation with `[, by = ]`, and update by reference, cross-referenced from tips 12 and 17.

Reply: We added he command from data.table as well.

Reviewer: - [Reviewer 3] A short note on when to leave base R for Rcpp or RcppArmadillo yourself, not only via a wrapper package.

Reply: This depends upon whether the use knows how to write in C++ in an efficient manner. So, we cannot advise everyone to do that, even if they can program in C++.

Recommended:

Reviewer: - [methods-soundness] Most matrix timing claims (for example crossprod versus t(X) %*% Y in Tips 52 to 53, princomp versus prcomp in Tip 46, and the float and Matrix tips) depend on the BLAS/LAPACK backend and the platform. Under an optimized multithreaded BLAS several of these gaps shrink or reverse. One sentence stating that the benchmarks were run with the reference BLAS on the described laptop, and that margins vary with the linear algebra backend, would set the reader's expectations correctly.

Reply: We added this sentence in the introduction "In the uploaded \textit{R} code we have added the session info of the \textit{R} environment we used. The LAPACK version is 3.12.1."

Reviewer: - [methods-soundness] Tip 64 says at most n eigenvalues of the correlation matrix are non-zero when n < p. After centering and standardizing, the rank is at most n - 1, so at most n - 1 eigenvalues are non-zero. Adjust the statement, since the surrounding argument about hd.eigen() rests on this rank.

Reply: We fixed this typo.

Reviewer: - [methods-soundness] Several timing displays (Tips 16 and 46, the profiling output in Tip 2) report microbenchmark or Rprof results, but the data-generating lines and any seed are given only for some. Adding the generating code and a set.seed for each reported benchmark would let a reader reproduce the numbers.

Reply: We do not want to add a .seed because the generated data are of not importance, the timing is. Also, you know that the time results will vary among computers. So, we simply state our timings, and the R code is uploaded as an easy way for the reader to run the code, instead of copying from the pdf of the paper, they can copy the code from the script easier.

Reviewer: - [statistical-rigor] Several quantitative speed claims carry no measurement: "at least 10 times faster" for Newton-Raphson versus nlm (tip 77), "3-4 times faster" for the G2 tests (tip 15), "more than 2 times faster" for sum(x %*% m) (tip 54), "almost 2 times faster"

for dim over nrow (tip 42), and the many "much faster" statements in Section 10. Either give the timing that supports each figure or state the number as an approximate observation on the authors' machine.

Reply: These are personal experience, empirical evidence. Consider this as something we have seen in general.

Reviewer: - [statistical-rigor] The two benchmark tables (tips 16 and 46) report min, lower quartile, mean, median, upper quartile and max over 20 replications but no reasoning for 20 runs and no statement of the run-to-run spread of the difference. For prcomp versus princomp the maxima (50.55 and 51.37 ms) nearly coincide while the medians differ by about 18 ms, so a reader cannot tell how stable the ordering is. Report the paired difference and its spread, and note that a single laptop with JIT and BLAS settings fixed limits how far the ratios generalise.

Reply: We report these because the R packages that measure time they give this type of output. We did not want to go deep in the comparison, memory for example, or what the reviewer suggests.

Reviewer: - [writing-quality] In the introduction, "Gillespie and Lovelace (2016) gave a bookâ€“length treatment in his book" refers to two authors, so "his" should read "their". The sentence also repeats "book" three times (Wickham "in his book", Burns "in his book The R Inferno", "a bookâ€“length treatment in his book"); one pass to vary the wording would read more cleanly.

Reply: We corrected this typo. We also proof read the whole manuscript again, after we completed the changes.

Reviewer: - [writing-quality] Tip 80 contains a spelling error, "afraind", that reads as an author typo rather than an extraction artefact. Please correct to "afraid".

Reply: We modified this sentence, and removed the verb "afraid" completely.

Reviewer: - [Reviewer 1] Quantify a representative set of the bare "faster" claims (for example tips 42, 54, 89) with the same benchmark script, so the factors ("almost 2 times", "more than 2 times faster", "10-50 times") are supported.

Reply: We cannot quantify because our observations differ between a laptop and a desktop. We cannot even quantify between our laptops, as we see different speed-ups.

Reviewer: - [Reviewer 2] Add collapse (fmean, fsum, fvar) to the grouped-operation tips (17, 38).

Reply: As we said in an earlier comment, and added in the Introduction, our comparisons take place against R's built-in functions. But, we do mention alternative packages as well.

Reviewer: - [Reviewer 3] Note the numerical-stability cost of `princomp` over `prcomp` in tip 46, consistent with the paper's own stability section.

Reply: We did not understand what the reviewer is trying to say with this comment.

Reviewer: - [Reviewer 3] Add a note on reproducible parallel RNG (`clusterSetRNGStream`, `mc.set.seed`) in Section 9, given the simulation use case the paper motivates.

Reply: We insist on not giving a .seed number, because the goal is to see the timings, not to reproduce exactly the same computations and mathematical results. The user may run these commands and see for themselves.

Reviewer: - [Reviewer 3] State the counter-case for the micro-optimizations that currently read as absolute (tip 42 `dim` vs `nrow`, tip 12 `mean` accuracy).

Reply: We did not understand what the reviewer is trying to say with their comment.

Optional:

Reviewer: - [methods-soundness] Tip 46 recommends princomp over prcomp on speed grounds, while the numerical-overflow section stresses stability. princomp via the covariance eigendecomposition is less accurate for ill-conditioned data than the SVD used by prcomp. A short note on when the speed gain is safe would align the two parts of the paper.

Reply: We were not aware of the stability issues to be honest and thank you for this. The goal of this paper is about mainly about speed. To us, what your are saying sounds like R's issue, something that R should fix.

Reviewer: - [statistical-rigor] A comparison across more than one machine, operating system and BLAS backend would show which of the reported ratios are properties of R and which are properties of this particular laptop.

Reply: When writing this paper we opted against this strategy, because perhaps more issues would arise and the paper would be lengthier. We list our tips, which we must highlight may not work across all laptops and as R evolves some may not work in the future. In the past for example and using another laptop we had seen ***that prcomp was faster than princomp****. In an earlier version of this paper, which can be found on the web, you will notice that we mention this.*

We will correct these in production, and you confirm them in the proof:

Reviewer: - [consistency-check] The profiling output in Tip 2 lists a function `"h"` (self.time 0.01) that does not appear in the function `f` shown just above it. The self-times still sum correctly (8.56 + 0.84 + 0.01 = 9.41), so the total is internally consistent, but the `"h"` row has no source in the displayed code. Confirm the intended output.

Reply: We ran it again and now present the new output.

Reviewer: - [writing-quality] Section 10 has a dropped package name: "The same is true for ordinal logistic regression in and in ordinal (Christensen, 2015)". A package name is missing before the first "in" (the comparison names only one of the two packages being compared). Please supply the intended package.

Reply: We corrected this typo.

Reviewer: - [Reviewer 1] The R Inferno (Patrick Burns) is cited in the introduction but has no entry in the references. Supply the reference.

Reply: We added the reference of the book.

Reviewer 1

****Recommendation:** minor**

This is a useful tutorial. The problem is real: R users who run simulations or matrix-heavy analyses routinely leave large speed and memory gains on the table, and a single well-organised list of concrete idioms has clear value for that audience. The coverage is broad and mostly accurate. The numerical-overflow section (tips 79 to 88) is the strongest part. The inverse-logit reformulation, the softplus formula max(x,0) + log1p(exp(-abs(x))), the log-sum-exp trick, working in log-densities, and using log.p for sortable tail probabilities are all correct and are exactly the things practitioners get wrong. The matrix-algebra section is also solid: crossprod/tcrossprod, solve(X, Y) over solve(X) %*% Y, backsolve/forwardsolve for triangular factors, and the associativity point in tip 61 are all sound and well chosen.

Reviewer: My main concern as an application reviewer is that many speed claims are asserted without evidence, and the few that carry benchmarks were run on a single laptop with an unstated BLAS. This matters because a good share of the matrix tips are BLAS-bound. Whether crossprod(X, Y) beats t(X) %*% Y, and by how much, and whether princomp beats prcomp, depends on whether R is linked against reference BLAS, OpenBLAS, or MKL. On an optimised multi-threaded BLAS some of these gaps shrink or reverse. Readers will apply these tips on machines unlike yours, so the paper should say which BLAS produced the numbers and should ship the script that produced tables 16 and 46. That also lets the quantified factors scattered through the text ("almost 2 times faster", "more than 2 times faster", "10-50 times faster") be checked. A short reproducibility script with sessionInfo would settle all of this at once and is the natural minimum for a tutorial whose value is its timings.

Reply: We have uploaded the R code and the sessionInfo() in the beginning of the code. The statements 10-50 times faster were observed in our laptops. Perhaps with a very strong desktop, and Linux instead of Windows, the speed-ups are smaller.

Reviewer: One consistency point deserves attention because the paper itself raises the standard. Section 7 states that one should not sacrifice numerical stability for speed. Tip 46

then recommends princomp over prcomp on speed grounds alone. princomp works from the covariance matrix and its eigendecomposition, which loses precision relative to the SVD of the centered data that prcomp uses, and the difference shows up on ill-conditioned or nearly collinear data that applied users meet often. A single sentence flagging the tradeoff keeps the tip while honouring the paper's own principle. A smaller correctness note: tip 36 gives x * (x > 0) for ifelse(x > 0, x, 0), which differs in the presence of NA, and tip 87's integer point is right and could add that length/dim/nrow return integers so n * n overflows silently, which the example already shows.

Reply: We were not aware of this limitation of princomp and so we added this sentence as a footnote in tip 46: "Note however, that \texttt{princomp} works from the covariance matrix and its eigendecomposition, which loses precision relative to the SVD of the centered data that \texttt{prcomp} uses, and the difference shows up on ill--conditioned or nearly collinear data that applied users might meet."

Reviewer: The headline count needs fixing. The abstract and introduction say "more than 100" tips, the title says "90+", and the list runs to 94. The claim that this is the first compilation of such length depends on getting that number straight, so please set it to what the paper actually contains. The R Inferno is discussed in the introduction but is absent from the references and should be added. On presentation, the tips read as a flat sequence within each section, and a few (38 on apply for medians, 48 on by) qualify their own advice in ways that could confuse a reader skimming for a rule. A one-line summary table grouping the tips by theme would help the tutorial's usability, though the paper stands without it.

Reply: We corrected the typo regarding the 100+ tips. An earlier version of the paper consisted of more than 70 tips. We expanded them to 100+ and then removed some of them, this is why this disagreement occurred, which we should have corrected. Bear in mind that these tips are based on our personal experience with our laptops using Windows.

Numbered suggestions to the authors:

Reviewer: 1. [required] Set the tip count consistently across the title, abstract, and introduction to the number actually present (94 by my count), since the "first such list" claim rests on it.

Reply: We now state in the Introduction that we have listed 94 tips.

Reviewer: 2. [required] State the BLAS used for the benchmarks and provide the script plus sessionInfo that produced the timing tables (tips 16 and 46). Note that crossprod vs explicit transpose-multiply, princomp vs prcomp, and the float throughput claim are BLAS-dependent and can narrow or invert under OpenBLAS or MKL, so present them as reference-BLAS results or qualify them.

Reply: We state in the Introduction the Lapack version used in our session.

Reviewer: 3. [required] Add one sentence to tip 46 noting that princomp is less numerically stable than prcomp because it eigendecomposes the covariance matrix, so the speed gain trades against precision on ill-conditioned data.

Reply: We added it.

Reviewer: 4. [editor_fix] Add a references entry for The R Inferno (P. Burns), which is cited in the introduction but missing from the reference list.

Reply: We added the reference.

Reviewer: 5. [recommended] Support the unquantified speed factors for a representative subset (for example tips 42, 54, and 89) using the same benchmark script, so the stated multipliers are backed by numbers.

Reply: These numbers are based on our experience and may not be observed with different machines or operating systems. What we want to stress is that these recommendations are faster, even to the slightest.

Reviewer: 6. [recommended] Note in tip 36 that x * (x > 0) and ifelse(x > 0, x, 0) differ when x contains NA, so readers do not swap them blindly.

Reply: We added this information under the chunk of the code.

Reviewer: 7. [optional] A short table grouping the tips by theme (memory, vectorization, matrix algebra, numerical stability) would make the tutorial easier to navigate and reference.

Reply: We like the grouping we have done, and that is why you see them organized by those sections.

Reviewer 2

****Recommendation:** major**

This is a tutorial, and I review it as one. A curated list of idioms does not owe a benchmark for every line the way a methods paper owes one for its headline result. The numerical-overflow section (tips 79 to 88) is the strongest part: the inverse-logit reformulation, the softplus identity max(x,0)+log1p(exp(-|x|)), the log-sum-exp shift, log1p and expm1, and working with log-densities are all correct and well chosen. The vectorization and preallocation advice is sound and the worked profiling example makes its point cleanly.

Reviewer: My concerns are methodological and they cluster. First, the paper is written by the maintainers of Rfast, Rfast2 and estats, and it recommends those packages as faster (tips 64 and 72, and Section 10). There is no competing-interest statement, and that needs to be added. Tied to it is a fairness question that runs through the paper. Many tips present a fast option against the slowest baseline only. Where the recommended option is an authored package, the relevant comparison is against the strongest alternative in the ecosystem, which for column medians, variances and distances means matrixStats, collapse and data.table, and for a truncated eigendecomposition (tip 64) means something like RSpectra or irlba. Please either benchmark against those or phrase the recommendation as an alternative to base R. I have marked the comparison required with the softening as the low-cost route.

Reply: Since the paper was based initially on comparing the three R packages you mentioned to the built-in R functions we decided to leave it as is, but mention alternative packages that offer fast executions as well. Regarding the statement, we will ask the editors to tell us what to add and where.

Reviewer: Second, reproducibility of the numbers you print. The paper shows two benchmark tables (centering in tip 16, princomp against prcomp in tip 46) and asserts ratios elsewhere ("more than 2 times faster", "almost 2 times faster"). None of the matrix timings can be read without the BLAS, since reference BLAS, OpenBLAS and MKL change the relative cost of crossprod, solve, eigen and the float tip (70) by large factors. The environment gives the R version but not the BLAS, and the code file cannot regenerate the two tables because

several chunks depend on objects defined only in the prose. A single script that builds the inputs, sets a seed, runs microbenchmark or bench, and prints the two tables would close this, or the printed numbers could be removed and the claims kept qualitative.

Reply: In the uploaded R code script you will see our sessionInfo(). There you will see the LAPACK version installed in our R environment.

Reviewer: Third, two technical corrections. Tip 84 states that p-values below .Machine$double.eps are rounded to zero. That is off by roughly 290 orders of magnitude. A p-value of 1e-100 is representable and sortable. Underflow to zero happens near the smallest normal double, about 1e-308, with subnormals below that. The advice to carry log p-values is good, so only the justification needs fixing. Tip 46 recommends princomp over prcomp for speed. princomp forms and diagonalises the covariance matrix, which is the less stable route, while prcomp uses the SVD of the centered data. In a paper whose own motto is not to trade stability for speed, that recommendation needs one sentence of caveat. In the same spirit, tip 12 recommends sum(x)/length(x) over mean for speed, and mean's extra cost buys the two-pass accuracy you are giving up, which is worth a half-sentence.

Reply: We have corrected the tip 84 now, and added a few worda about tip 46. Regarding tip 12, the accuracy is mentioned.

Reviewer: A lighter point on the many micro-claims. "nrow is almost 2 times faster than dim(X)[1]" (tip 42), the internal .colSums (tip 39), iterating over elements (tip 93): these are true as ratios and negligible as absolute times. A reader can spend real effort chasing nanoseconds. Where you give a ratio without a measurement, either measure it or mark it as a constant-factor nicety.

Reply: It is true, the savings is measured in nanoseconds. But, for someone who performs millions of simulation studies every week, the nanoseconds become seconds or minutes with time. The message we want to pass is that one should make light-speed fast functions whenever possible. If a model requires 0.01 seconds to fit once, when it comes to millions of repeats, the time cost is high.

Numbered suggestions to the authors:

Reviewer: 1. [required] Add a competing-interest statement naming Rfast, Rfast2 and estats as packages the authors maintain, since the paper recommends them (tips 64, 72, Section 10).

Reply: We will ask the editors what to do with this, how to write the statement.

Reviewer: 2. [required] For each authored-package recommendation, compare against the strongest alternative (matrixStats, collapse, data.table, and RSpectra or irlba for tip 64), or reword the claim to "an alternative to base R" where you do not benchmark.

Reply: Our initial decision was to compare against the built-in functions for illustration purposes. Overall, the goal is to show the users how to write efficient code. To avoid the comparison we mention a few packages and encourage authors to search for fast alternatives in packages.

Reviewer: 3. [required] Supply a runnable script that regenerates the tables in tips 16 and 46 with a fixed seed, the R version, and the BLAS stated, or remove the printed timings and keep the claims qualitative.

Reply: We have uploaded the updated R code with the session info included where we show the LAPACK version in our R environment.

Reviewer: 4. [required] In tip 84, replace the .Machine$double.eps threshold with the correct underflow behaviour near 1e-308 and the subnormal range, keeping the log.p recommendation.

Reply: We fixed this.

Reviewer: 5. [required] Add one sentence to tip 46 that princomp diagonalises the covariance matrix and is less stable than the SVD-based prcomp, so the speed gain carries an accuracy cost.

Reply: We added a sentence regarding the computational instability the princomp may face in contrast to the more robust prcomp.

Reviewer: 6. [editor_fix] Make the tip count consistent: the abstract says "more than 100", the title says "90+", and the list ends at 94.

Reply: We fixed this typo.

Reviewer: 7. [recommended] Add collapse (fmean, fsum, fvar) to the grouped-operation tips 17 and 38, where it is typically faster than rowsum and aggregate.

Reply: We preferred to compare our functions against the R's built-in functions. We do mention this and other packages in our manuscript.

Reviewer: 8. [recommended] Temper the unquantified ratio claims (tips 42, 54, 70, 93) to qualitative statements, or back each with a measurement, and add a half-sentence to tip 12 on the accuracy that sum/length gives up relative to mean.

Reply: The speed-ups are based on our personal experience, and vary between laptops and desktops, this is why we do not report the exact numbers.

Reviewer: 9. [optional] A table of the matrix-tip speedups under reference BLAS, OpenBLAS and MKL would settle how much each recommendation depends on the linear algebra backend.

Reply: We have LAPACK in our R environment.

Reviewer 3

I read this as a software and reproducibility reviewer, with the uploaded R file treated as part of the submission. The tutorial is useful and most of the advice is correct and well chosen. The matrix-algebra tips (crossprod, solve(X, Y), backsolve, the Mahalanobis route to the multivariate normal exponent) and the numerical-overflow section (log-sum-exp, log1p, log-densities, log p-values) are the strongest parts and are the kind of thing many R users genuinely do not know. My concerns are about whether the shipped artifact does what the submission record says it does, one correctness bug, and a disclosure that a computing venue expects.

Reviewer: The declared entry point is `Rscript R code.R`, which signals a runnable artifact. It is not one. The file's own header states that several fragments refer to objects defined only in surrounding prose, and a top-to-bottom run fails early: chunk 2 calls `microbenchmark(mean(x), ...)` with no `x` defined, chunk 5 uses `n`, chunk 6 uses `fast` and `slow`, and so on. No `set.seed` appears anywhere, so even the self-contained chunks that build data with `rnorm` do not reproduce fixed numbers. The two timing tables in the paper (centering in tip 16, princomp against prcomp in tip 46) are pasted into the code as comments and are not regenerated by any runnable, seeded code. The machine is named, which is good provenance, but timings also depend on the BLAS, the OS and the R build, none of which is pinned beyond the prose. For a paper whose entire value rests on speed claims, a reader should be able to run something and see the numbers appear. The cheapest honest fix is to split the file: keep the illustrative snippets as clearly non-executable, and add two or three small seeded scripts that build the inputs and print the benchmark tables. If the file is meant to be snippets only, the declared entry point should say so.

Reply: We prefer to give the script with all snippets of code that appear in the manuscript. The reviewer has a point, x does not appear, but we mention the length or dimensions of x (if matrix) in the manuscript. The updated uploaded R code.R script includes now the sessionInfo() where the user can see that we use LAPACK in our R environment.

Reviewer: Tip 17 has a correctness bug in its "a bit faster" variant. `rowsum(x, ina, reorder = FALSE)` returns group rows in order of first appearance, while `as.vector(table(ina))` returns counts in sorted factor-level order. When the two orders differ, each group mean is divided by another group's count. The default `rowsum(x, ina)` in a2 is fine because it sorts. Please

either align the counts to the `reorder = FALSE` order or drop a3. This belongs in a tutorial precisely because a reader will copy it.

Reply: We fixed this typo.

Reviewer: The authors maintain Rfast, Rfast2 and estats, and the paper recommends these packages in tips 64, 72 and 74 and across Sections 9 and 10. A reader is entitled to know that. Please add a short interest declaration. This is not a judgement on the recommendations, several of which are reasonable, only on disclosing who wrote the packages being recommended.

Reply: We will ask the editors what to write and where to place it.

Reviewer: Two smaller content points are worth lining up with the paper's own stated priorities. Section 7 argues that numerical stability should not be sacrificed for speed, yet tip 46 recommends `princomp` over `prcomp` purely on timing. `princomp` forms the covariance and eigendecomposes it, which is less accurate for near-collinear data than the SVD route in `prcomp`, and one sentence should say so. On data.table, the paper uses it only through `fread`. For the aggregation tips (12 and 17) and for large grouped work generally, `[, by = ]`, keyed joins and update by reference are often the fastest route available, and a tutorial on efficient R aimed at large data reads as incomplete without them. The tip count in the abstract and introduction ("more than 100") does not match the 94 numbered tips or the "90+" in the title.

Reply: We have resolved these issues.

Numbered suggestions to the authors:

Reviewer: 1. [required] Make `R code.R` run end to end with inputs defined and `set.seed` set, or re-declare it as non-executable snippets and add separate seeded scripts that regenerate the centering (tip 16) and princomp/prcomp (tip 46) tables.

Reply: We prefer to avoid the .seed number, as we are not interested in the mathematical results, but on the timing results.

Reviewer: 2. [required] Fix tip 17: align `rowsum(x, ina, reorder = FALSE)` with counts in the same order, or remove the a3 variant, because dividing by `as.vector(table(ina))` uses sorted-order counts.

Reply: We fixed this inconsistency.

Reviewer: 3. [required] Add a declaration of the authors' interest in Rfast, Rfast2 and estats where those packages are recommended.

Reply: We will ask the editor's opinion on how to address this.

Reviewer: 4. [editor_fix] Reconcile the tip count: abstract and introduction say "more than 100", the list ends at 94, and the title says "90+".

Reply: We fixed it.

Reviewer: 5. [recommended] Add data.table beyond `fread`: keyed joins, grouped aggregation with `[, by = ]`, and update by reference, linked from tips 12 and 17.

Reply: We did not understand the reviewer's comment here.

Reviewer: 6. [recommended] In tip 46, state that `princomp` is less accurate than `prcomp` for near-collinear data.

Reply: We added this information.

Reviewer: 7. [recommended] In Section 9, add a note on reproducible parallel RNG using `clusterSetRNGStream` and `mc.set.seed`.

Reply: We prefer not to have a .seed number, as our prime interest lies on time, not on reproducing the same mathematical computations.

Reviewer: 8. [recommended] Give the counter-case for the micro-optimizations that read as absolute, including tip 42 (`dim` vs `nrow`) and the accuracy cost in tip 12 (`mean` vs `sum(x)/length(x)`).

Reply: These are based on personal experience, so the seed-ups may differ, and we prefer not to state them.

Reviewer: 9. [optional] Add a short note on when to move to Rcpp or RcppArmadillo yourself, as a complement to the wrapper-package route.

Reply: This depends upon whether the use knows how to write in C++ in an efficient manner. So, we cannot advise everyone to do that, even if they can program in C++.