TABLE I

COMPLETE LIST OF ANONYMISED ANSWERS TO QUESTION 10: "GIVEN THE AVAILABILITY OF TESTING TOOLS FOR APP DEVELOPMENT, IN YOUR OPINION WHAT ARE THE TOP 2 THINGS YOU LOOK FOR/NEED/WOULD LIKE TO SEE?". GRAMMAR HAS NOT BEEN CURATED.

Replies
Shorter dev cycles time between updating package/adding test cases, and the getting the results of those tests; better IDE support
Able to write tests quickly; able to run tests easily
Honestly anything that would make it faster or automate bits of it. I do a terrible job of testing, and it's largely because it just takes so much time for so little immediate benefit.
I haven't had any problem with `testthat`
Free deploy and testing, stats on usage
I don't really understand the question.
test database support, better organization
Granular reporting and consistent testing environments
Test parametrization; "clever" visual testing that tests visual view and not picture file
Mocking
Testing profiles, integration with CI/CD
better integration with CI, fewer transitive dependencies
tutorials
nothing missing
More education on the need for testing
better educational resources to learn about developing good tests for various purposes (e.g., regression tests, unit tests, etc)
I like the describe/it style unit testing syntax. `testthat` does have this style (https://rdrr.io/cran/testthat/man/describe.html), but not well developed.
easy automatic testing of user interface such as shiny
Better tools for investigating test failures
documentation, elegance
Less dependencies on other packages piling up, more dependency on base R tools.
Monetary incentives for maintaining and testing tools
Better integration of test results into CI processes and treating test failures as data not an error that causes CI to fail.
Easier transition from informal tests used during development to formal tests once the package is more mature.
Easy to generate test samples; Easy to test if the result is right
1- A document that would explain how to create meaningful/efficient unit tests (i.e. beyond just boosting the code coverage). 2- I only use `testthat`, may be a comparison with the different testing packages would be helpful.
Easier automated input type testing
good examples of tests, learning resources
simplicity & flexibility
Better documentation (especially tutorials, examples and prebuilt code templates); wider community support
more ubiquitous training in their use.
Better way of looping tests for test cases; easy way to test on other operating systems
More consistent online testing services that run the current R development release on all supported platforms
Integration with CI tools (Travis-CI, Appveyor, etc.) with Github hooks is nice. For some of my packages `covr`'s `codecov` will error in a timeout in a CI environment so I need to remember to manually run it locally.
The automation of test generation. This would be helpful in incorporating tests that are standard across different packages.
Integration with R CMD check
easy to use
for R specifically, easier access to running CRAN's checks in many environments / architectures would help a lot
parallel testing, better documented best practices
more diverse set of testing tools
Code performance, Compatibility
Understanding of how to build tests for data science (and data tables generally).
readable reports, good compatibility with CI and RStudio
Easier testing of plotting code
find potential bugs, testing functionality under different OS
easy to create units test, integrated in RStudio/`devtools`
packages that help test wrappers and interfaces to external APIs
(1) testing tools should not be more burdensome than standard (i.e., R CMD check) approaches. (2) some kind of conversion tool to convert existing tests to the new format (i.e., to do for testing what `roxygenize` does for documentation)
Easier ways to generate test data that's small and easy to include in a package, easy documentation of test result changes to track function changes over time in a documented way.
code coverage, automatic generation of common tests
ease of use, good documentation
faster ways to diagnose and re-test specific test cases across many platforms (e.g. if a test case fails on one platform using Travis or Github actions, being able to re-run just that test case quickly rather than having to re-run the entire test infrastructure)
ease of use; good documentation