SOFTWARE TOOL ARTICLE

REVISED testCompareR: an R package to compare two binary diagnostic tests using paired data

[version 4; peer review: 1 approved, 2 approved with reservations]

Kyle J. Wilson[1,2], José A. Roldán-Nofuentes[3], Marc Y.R. Henrion[2,4]

[1]University of Liverpool, Liverpool, L7 8TX, UK
[2]Malawi-Liverpool-Wellcome Trust Clinical Research Programme, Blantyre, Southern Region, Malawi
[3]Universidad de Granada, Granada, Andalusia, 18010, Spain
[4]Liverpool School of Tropical Medicine, Liverpool, L3 5QA, UK

v4 **First published:** 02 Jul 2024, **9**:351 https://doi.org/10.12688/wellcomeopenres.22411.1
Second version: 11 Oct 2024, **9**:351 https://doi.org/10.12688/wellcomeopenres.22411.2
Third version: 21 Oct 2024, **9**:351 https://doi.org/10.12688/wellcomeopenres.22411.3
Latest published: 26 Nov 2024, **9**:351 https://doi.org/10.12688/wellcomeopenres.22411.4

Abstract

Background

Binary diagnostic tests are commonly used in medicine to answer a question about a patient's clinical status, most commonly, do they or do they not have some disease. Recent advances in statistical methodologies for performing inferential statistics to compare commonly used test metrics for two diagnostic tests have not yet been implemented in a statistical package.

Methods

Up-to-date statistical methods to compare the test metrics achieved by two binary diagnostic tests are implemented in the new R package testCompareR. The output and efficiency of testCompareR is compared to the only other available package which performs this function, DTComPair, as well as an open-source program, compbdt, using a motivating example.

Results

testCompareR achieves similar results to DTComPair using statistical methods with improved coverage and asymptotic performance. Further, testCompareR is faster than the currently available package

Open Peer Review

Approval Status ✓ ? ?

	1	2	3
version 4 (revision) 26 Nov 2024			
version 3 (revision) 21 Oct 2024	✓ view		? view
version 2 (revision) 11 Oct 2024			
version 1 02 Jul 2024	? view	? view	

1. **Dennis Robert**, Qure.ai, Bangalore, India
2. **Aaron Caldwell**, University of Arkansas for Medical Sciences, Arkansas, USA
3. **Haydar Demirhan**, RMIT University, Melbourne, Australia

Any reports and responses or comments on the article can be found at the end of the article.

and requires fewer pre-processing steps in order to produce accurate results.

Conclusions

testCompareR provides a new tool to compare the test metrics for two binary diagnostic tests compared with the gold standard. This tool allows flexible inputs, which minimises the need for data pre-processing, and operates in very few steps, so that it is easy to use even for those less experienced with R. testCompareR achieves results comparable to those computed by DTComPair, using optimised statistical methods and with improved computational efficiency.

Plain Language Summary

testCompareR is a new package for the statistical programming language R which compares the performance of two binary diagnostic tests. Binary diagnostic tests are tests which give either a positive or negative outcome. A good example is the lateral flow test commonly used to diagnose COVID-19, but they are widely used in medicine.

testCompareR is faster and more efficient than existing options like DTComPair, with comparable accuracy and fewer pre-processing requirements. This means it's easier to use, especially for those new to R. testCompareR is a valuable option for researchers and clinicians needing to compare test metrics from two binary diagnostic tests.

Keywords
R package, diagnostic test, paired data, dichotomous, binary, compare

Corresponding author: Kyle J. Wilson (kyle.wilson@liverpool.ac.uk)

Author roles: Wilson KJ: Conceptualization, Methodology, Software, Validation, Writing – Original Draft Preparation, Writing – Review & Editing; **Roldán-Nofuentes JA**: Methodology, Software, Validation, Writing – Review & Editing; **Henrion MYR**: Supervision, Validation, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

Grant information: This work was supported in part by Wellcome [223502, 222530 (KJW) and 206545 (MYRH)]. This study was partially funded by Spanish Ministry of Science and Innovation, Grant “PID2021-126095NB-100” funded by MCIN/AEI/10.13039/501100011033 and by “ERDF A way of making Europe” (JARN).
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

How to cite this article: Wilson KJ, Roldán-Nofuentes JA and Henrion MYR. **testCompareR: an R package to compare two binary diagnostic tests using paired data [version 4; peer review: 1 approved, 2 approved with reservations]** Wellcome Open Research 2024, **9**:351 https://doi.org/10.12688/wellcomeopenres.22411.4

First published: 02 Jul 2024, **9**:351 https://doi.org/10.12688/wellcomeopenres.22411.1

REVISED **Amendments from Version 3**

This version of the manuscript includes corrections and improvements suggested during peer review. Major changes in response to Haydar Demirhan's excellent feedback include a table to summarise the features included in test CompareR, DTComPair and compbdt and updates to both the manuscript and the benchmarking supplement (which is available on Github and archived on Zenodo) to include information about computation time as the size of the data set increases. Responses to the reviewers has been addressed in previous versions.

Any further responses from the reviewers can be found at the end of the article

Introduction

The determination of disease status based upon some diagnostic test is a fundamental principle in medicine. Tests may be straightforward, for example the presence or absence of crepitations on lung auscultation, or highly complex, such as the identification of specific changes in a patient's genetic code, but very often clinicians are seeking to answer a simple question with a dichotomous answer: does this patient have or not have the disease in question?

Accordingly, very many tests have been developed which seek to provide a simple and interpretable binary result, either by identifying a target which is disease specific and the presence or absence of which confirms or refutes the diagnosis, or by providing some threshold value above (or below) which the patient can be considered positive for the disease[1–3].

Binary diagnostic tests such as these rarely, if ever, perform perfectly. During development, diagnostic tests are often compared to a gold standard; a reference test or clinical diagnosis which defines true disease status for an individual. From this we can derive the fundamental performance characteristics, such as sensitivity, specificity, positive and negative predictive values and likelihood ratios. As for any other estimated quantities, principled comparison of these test metrics for two different tests requires the use of statistical inference.

Although the comparison of test metrics has been the subject of much academic inquiry, to the best of our knowledge, there is only one package for the open-source statistical programming language R[4] which performs this function. The `DTComPair` package[5] uses well-established statistical methods to perform statistical inference when comparing test metrics. The package is available from the Comprehensive R Archiving Network (CRAN).

A newer program, `compbdt`, has also been published in an open-access journal[6]. This program uses the most up-to-date statistical methods. However, the code is presented as one large function and is therefore unavailable on CRAN. This limits the useability of the program, as users are required to search the statistical literature and be sufficiently proficient with R to import and run the function.

We sought to develop a new R package which performs both descriptive and inferential statistics for the commonly used test metrics, combining the optimised statistical methods of `compbdt` with the useability and availability of `DTComPair`.

Because the target users of this package are clinicians involved in the development and evaluation of diagnostic tests, not statisticians or computational scientists, we defined a list of features to maximise usability. Specifically, the new package is designed to:

- Take a data frame or matrix as an argument containing all commonly used binary operators (eg. yes/no, y/n, pos/neg, 1/0, etc.).
- Return output following a single function call.
- Display a contingency table (confusion matrix) summarising the raw data.
- Allow users to select whether this matrix has margins displaying row and column sums.
- Return the prevalence of the condition in question (according to the reference standard) and a confidence interval based on the cohort studied.
- Allow the user to select which pairs of test metrics they are interested in (e.g. sensitivity/specificity) and exclude those which are not relevant to their hypothesis.
- Return a matrix for each selected test metric displaying point estimates for both tests, alongside standard errors and confidence intervals.
- Return test statistics and p-values for differences between the selected test metrics for each of the two tests

- Handle multiple testing using standard correction methods.
- Offer the user the option of continuity correction if McNemar's test is indicated.
- Allow the user to input test names to facilitate interpretation.
- Provide an optional function which interprets the output (in plain English) for the user.
- Provide an additional function for summarising descriptive statistics for one test.

Presence of absence of these features has been summarised for each of the available packages and programs in Table 1.

Here we introduce `testCompareR`, a new R package which calculates the performance metrics for two diagnostic tests by comparing them against a user-provided gold standard test, then compares the performance metrics of those two binary diagnostic tests to one another, all subject to a paired experimental design.

Methods

Implementation

Statistical methods

Calculating the test metrics

Each of the test metrics is calculated using well-established and standardised formulas[7–9], based upon standard contingency tables comparing the gold standard and the test results (Table 2).

Diagnostic accuracies (sensitivity and specificity)

$$Se = \frac{TP}{TP + FN} \qquad Sp = \frac{TN}{TN + FP}$$

Table 1. Summary of features by package / program. i) testCompareR permits user to select which tests to perform via Boolean arguments to the compareR function. DTComPair achieves the same through manual selection of which functions to perform. ii) testCompareR allows users to correct for multiple comparisons using any of the p.adjust.methods from the R stats package. compbdt uses Holm correction by default. Changing this requires the user to manually edit the function.

Feature	testCompareR	DTComPair	compbdt
Provide data as a data frame	Y	N	N
Flexible input for positive and negative values	Y	N	N
Perform analysis in a single function call	Y	N	N
Display contingency tables	Y	Y	N
User selection of pairs of test metrics	Y[i]	Y[i]	N
Point estimates and confidence intervals	Y	Y	Y
Test statistics and p values	Y	Y	Y
Correction for multiple comparisons	Y[ii]	N	Y[ii]
Option of continuity correction	Y	Y	N
User-defined test names	Y	Y	N
Summarise descriptive statistics for one test	Y	Y	N

Table 2. Contingency table evaluating a test against a gold standard.

		Test	
		+	-
Gold standard	+	True positive (TP)	False negative (FN)
	-	False positive (FP)	True negative (TN)

Predictive values

$$PPV = \frac{TP}{TP + FP} \qquad NPV = \frac{TN}{TN + FN}$$

Likelihood ratios

$$PLR = \frac{Se}{1 - Sp} \qquad NLR = \frac{1 - Se}{Sp}$$

Estimating confidence intervals

The diagnostic accuracies and predictive values are binomial proportions, for which several methods exist to estimate confidence intervals. Yu *et al.* (2014, see 10) proposed a modification of the Wilson interval and demonstrated superior performance compared to other commonly used intervals.

The likelihood ratios are not binomial proportions, but rather ratios of two independent binomial proportions. A comprehensive review and simulation of methods to estimate confidence intervals for the ratios of binomial proportions demonstrated that an approximation to the score method had superior performance[11].

These methods are implemented within the testCompareR package. For detailed mathematical descriptions, see ‘Mathematical descriptions’ at the end.

Hypothesis testing

Diagnostic accuracies

Simulation studies have demonstrated that the best methods for comparing diagnostic accuracies obtained from paired data vary depending on prevalence and total number of participants[12].

As a rule of thumb, in cases where prevalence is low (<10%) and the total number of participants is less than 100, the Wald test should be used to test two null hypotheses[12]:

$$H_0 : Se_1 = Se_2$$

$$H_0 : Sp_1 = Sp_2$$

Where either condition remains unmet the optimal method involves first testing the global null hypothesis:

$$H_0 : Se_1 = Se_2 \text{ and } Sp_1 = Sp_2$$

$$H_1 : Se_1 \neq Se_2 \text{ or } Sp_1 \neq Sp_2$$

The Wald test statistic forms the basis of this test. If the global null hypothesis is not rejected then neither individual null hypothesis should be considered unmet. When the global null is rejected then individual hypothesis tests are performed to determine whether the sensitivities are significantly different, or the specificities or both. When the total number of participants is less than or equal to 100, or greater than or equal to 1000, then the Wald test statistic performs best according to Roldán-Nofuentes and Sidaty-Regad[12]. In cases where total number of participants is between 100 and 1000 then McNemar’s test should be used[12]. In the testCompareR package, McNemar’s test is performed with continuity correction by default. A decision tree summarising this process is shown in Figure 1.

Predictive values

In a manner similar to that seen for diagnostic accuracies, the approach to hypothesis testing for the predictive values relies upon the Wald test statistic to first perform a global hypothesis test[13]. If the global hypothesis is rejected, the causes of significance are investigated using a weighted generalised score statistic, as described by Kosinski[14].

Likelihood ratios

The `testCompareR` package also uses global hypothesis testing to compare the likelihood ratios. The global hypothesis test considers the natural logarithm of the ratios of the positive likelihood ratios and negative likelihood ratios before calculating the Wald test statistic[15]. Where the global null is rejected the cause of significance is determined by applying the same statistical methods individually.

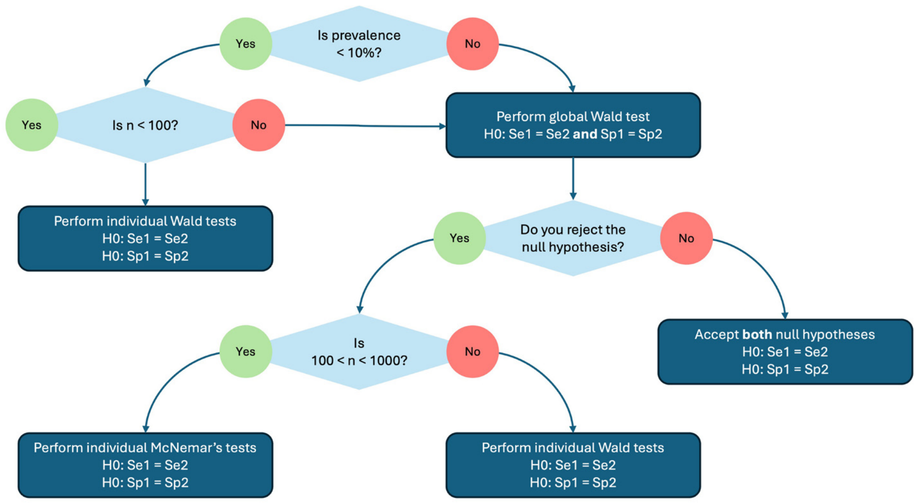
Is prevalence < 10%?
Yes
No
Is n < 100?
Yes
No
Perform global Wald test H0: Se1 = Se2 and Sp1 = Sp2
Perform individual Wald tests H0: Se1 = Se2 H0: Sp1 = Sp2
Do you reject the null hypothesis?
Yes
No
Accept both null hypotheses H0: Se1 = Se2 H0: Sp1 = Sp2
Is 100 < n < 1000?
Yes
No
Perform individual McNemar's tests H0: Se1 = Se2 H0: Sp1 = Sp2
Perform individual Wald tests H0: Se1 = Se2 H0: Sp1 = Sp2

Figure 1. A decision tree showing how the hypothesis testing for diagnostic accuracies (sensitivity and specifity) is performed by the testCompareR package.

Installation

`testCompareR` is available from CRAN and can be installed via the `install.packages()` function. This version should be the preferred version for most users. The development version with the most current features is available from GitHub.

```
# install from CRAN
install.packages("testCompareR")

# install development version
if(require("devtools")) {
  install_github("kajlinko/testCompareR")
} else {
  install.packages("devtools")
  require("devtools")
  install_github("kajlinko/testCompareR")
}
```

Data preparation

Flexible data entry is one of the key features of `testCompareR`. This minimises the number of pre-processing steps required by the user. In fact, for users not proficient with R, pre-processing could be handled entirely within spreadsheet or database software. There are only two steps which are imperative.

Firstly, positive and negative results must be coded according to a list of acceptable values. This list is relatively extensive, incorporating commonly used synonyms for coding positive and negative results in the English language (see Table 3). There is no requirement for consistency, which may benefit researchers performing secondary data analyses using data collated from multiple sources. Additionally, the package handles cases and white space so that researchers do not have to manually or computationally re-code their data.

Table 3. List of acceptable values when coding data from binary diagnostic tests. Values in inverted commas indicate character strings. Integer values are denoted without quotation marks. The acceptable values are not case sensitive, e.g. “pos”, “Pos”, “POS” would all be acceptable for positive result coding.

	List of acceptable values
Positive	"positive", "pos", "p", "yes", "y", "+", "1", "true", "t", 1
Negative	"negative", "neg", "no", "n", "-", "0", "2", "false", "f", 0, 2

Secondly, the structure of the data as presented to the package must conform to two rules:

1) The data need to be paired, i.e. the results on each row are for the same test sample or individual.
2) The observations on each individual row are independent, i.e. no biological or technical replicates, which violate the assumption of independence.

Data can either be provided as a data frame (to the `df` argument) or as a series of vectors (to the `test1`, `test2` and `gold` arguments. Users who supply a data frame to the `df` argument may choose to specify the columns which should be used for analysis, by passing strings or integers to `test1`, `test2` and `gold`. In the case where columns are not specified compareR will default to columns one, two and three for `test1`, `test2` and `gold`, respectively, while issuing a warning to inform the user to check that this is correct. Failure to comply with rules 1 and 2 may produce sensible-looking results which do not answer the question asked by the researcher. Users should therefore take extra care to ensure their data are appropriate for this method before implementing their analysis.

`testCompareR` currently expects complete data to be provided to the functions. The user is required to decide how to deal with missing values (e.g. complete case analysis, single value imputation, multiple imputation). If in doubt, users of the package should discuss their individual situation with an experienced statistician.

Statistical descriptions

Here we describe the mathematical equations for each of the confidence intervals calculated in the testCompareR paper.

First, we must define the contingency table which compares both tests under evaluation against the gold standard. This defines the fundamental values which will be used in subsequent calculations (see Table 4).

Sensitivity (Yu *et al.* interval):

$$1.\quad 0.5+\frac{s+z^4_{1-\alpha/2}/53}{s+z^4_{1-\alpha/2}}(\hat{S}e_i-0.5)\pm\frac{z_{1-\alpha/2}}{s+z^2_{1-\alpha/2}}\sqrt{s(1-\hat{S}e_i)\hat{S}e_i+\frac{z^2_{1-\alpha/2}}{4}}$$

where $z_{1-\alpha/2}$ is the 100(1 – α/2)th percentile of a standard normal distribution, s represents the total number of positive cases (ie. $s_{11}+s_{10}$ for test 1 and $s_{11}+s_{01}$ for test 2) and Se_i represents the point estimate of sensitivity for test 1 or test 2.

Specificity (Yu *et al.* interval):

$$2.\quad 0.5+\frac{r+z^4_{1-\alpha/2}/53}{r+z^4_{1-\alpha/2}}(\hat{S}p_i-0.5)\pm\frac{z_{1-\alpha/2}}{r+z^2_{1-\alpha/2}}\sqrt{r(1-\hat{S}p_i)\hat{S}p_i+\frac{z^2_{1-\alpha/2}}{4}}$$

where $z_{1-\alpha/2}$ is the 100(1 – α/2)th percentile of a standard normal distribution, r represents the total number of negative cases (ie. $r_{00}+r_{01}$ for test 1 and $r_{00}+r_{10}$ for test 2) and Sp_i represents the point estimate of specificity for test 1 or test 2.

Positive predictive value (Yu *et al.* interval):

$$3.\quad 0.5+\frac{n_{1\cdot}+z^4_{1-\alpha/2}/53}{n_{1\cdot}+z^4_{1-\alpha/2}}(P\hat{P}V_1-0.5)\pm\frac{z_{1-\alpha/2}}{n_{1\cdot}+z^2_{1-\alpha/2}}\sqrt{n_{1\cdot}(1-P\hat{P}V_1)P\hat{P}V_1+\frac{z^2_{1-\alpha/2}}{4}}$$

Table 4. Table of fundamental values. Each value represents the number of individuals meeting the conditions. For example, s11 represents the number of individuals for whom the gold standard, Test 1 and Test 2 are all positive.

		Test 1 +		Test 1 -		Column totals
		Test 2 +	Test 2 -	Test 2 +	Test 2 -	
Gold standard	+	s11	s10	s01	s00	ss
	-	r11	r10	r01	r00	rr
Row totals		n11	n10	n01	n00	n

$$4. \quad 0.5 + \frac{n_{\cdot 1} + z_{1-\alpha/2}^4/53}{n_{\cdot 1} + z_{1-\alpha/2}^4}(P\hat{P}V_2 - 0.5) \pm \frac{z_{1-\alpha/2}}{n_{\cdot 1} + z_{1-\alpha/2}^2}\sqrt{n_{\cdot 1}(1 - P\hat{P}V_2)P\hat{P}V_2 + \frac{z_{1-\alpha/2}^2}{4}}$$

where $n_{1\cdot} = n_{11} + n_{10}$ and $n_{\cdot 1} = n_{11} + n_{01}$.

Negative predictive value (Yu *et al.* interval):

$$5. \quad 0.5 + \frac{n_{0\cdot} + z_{1-\alpha/2}^4/53}{n_{0\cdot} + z_{1-\alpha/2}^4}(N\hat{P}V_1 - 0.5) \pm \frac{z_{1-\alpha/2}}{n_{0\cdot} + z_{1-\alpha/2}^2}\sqrt{n_{0\cdot}(1 - N\hat{P}V_1)N\hat{P}V_1 + \frac{z_{1-\alpha/2}^2}{4}}$$

$$6. \quad 0.5 + \frac{n_{\cdot 0} + z_{1-\alpha/2}^4/53}{n_{\cdot 0} + z_{1-\alpha/2}^4}(N\hat{P}V_2 - 0.5) \pm \frac{z_{1-\alpha/2}}{n_{\cdot 0} + z_{1-\alpha/2}^2}\sqrt{n_{\cdot 0}(1 - N\hat{P}V_2)N\hat{P}V_2 + \frac{z_{1-\alpha/2}^2}{4}}$$

where $n_{0\cdot} = n_{00} + n_{01}$ and $n_{\cdot 0} = n_{00} + n_{10}$.

Positive likelihood ratio (approximation to the score method):

$$7. \quad \frac{\tilde{n}\tilde{s}_{1\cdot}\tilde{r}_{1\cdot} + \frac{z_{1-\alpha/2}^2}{2}(\tilde{s}\tilde{s}_{1\cdot} + \tilde{r}\tilde{r}_{1\cdot}' - 2\tilde{s}_{1\cdot}\tilde{r}_{1\cdot}) \pm z_{1-\alpha/2}\sqrt{\tilde{n}^2\tilde{s}_{1\cdot}\tilde{r}_{1\cdot}\left[\tilde{s}_{1\cdot} + \tilde{r}_{1\cdot} - \tilde{n}\tilde{S}e_1(1 - \tilde{S}p_1)\right] + \frac{z_{1-\alpha/2}^2}{4}(\tilde{s}\tilde{s}_{1\cdot} - \tilde{r}\tilde{r}_{1\cdot})^2}}{\tilde{r}_{1\cdot}\left[\tilde{n}\tilde{s}(1 - \tilde{S}p_1) - z_{1-\alpha/2}^2(\tilde{s} - \tilde{r}_{1\cdot})\right]}$$

where $\tilde{s}_{1\cdot} = s_{1\cdot} + 0.5$, $\tilde{r}_{1\cdot} = r_{1\cdot} + 0.5$, $\tilde{s} = s + 1$, $\tilde{r} = r + 1$, $\tilde{n} = n + 2$, $\tilde{S}e_1 = \tilde{s}_{1\cdot}/\tilde{s}$ and $\tilde{S}p_1 = \tilde{r}_{0\cdot}/\tilde{r}$.

Regarding PLR_1, if the lower limit of the confidence interval is less than $\tilde{s}_{1\cdot}/(\tilde{n} - \tilde{r}_{1\cdot})$ or greater than $\widehat{PLR}_1$ then the lower limit is given by:

$$8. \quad \frac{\tilde{s}_{1\cdot}(1 - \tilde{S}p_1) + \frac{z_{1-\alpha/2}^2}{2} - z_{1-\alpha/2}\sqrt{\frac{z_{1-\alpha/2}^2}{4} + \tilde{s}_{1\cdot}(1 - \tilde{S}p_1 - \tilde{S}e_2)}}{\tilde{S}(1 - \tilde{S}p_1)^2 + z_{1-\alpha/2}^2}$$

Equally, if the upper limit is greater than $(\tilde{n} - \tilde{s}_{1\cdot})/\tilde{r}_{1\cdot}$ or less than $\widehat{PLR}_1$ then the upper limit is given by:

$$9. \quad \frac{\tilde{r}_{1\cdot}\tilde{S}e_1 + \frac{z_{1-\alpha/2}^2}{2} + z_{1-\alpha/2}\sqrt{\frac{z_{1-\alpha/2}^2}{4} + \tilde{r}_{1\cdot}(\tilde{S}e_1 + \tilde{S}p_1 - 1)}}{\tilde{r}(1 - \tilde{S}p_1)^2}$$

Replacing $x_{1\cdot}$ with $x_{\cdot 1}$ where x represents any of the fundamental values s, r or n and $\tilde{S}e_1$ and $\tilde{S}p_1$ with $\tilde{S}e_2$ and $\tilde{S}p_2$, respectively, will return the values for $\widehat{PLR}_2$.

Negative likelihood ratio (approximation to the score method):

$$10. \quad \frac{\tilde{n}\tilde{s}_{0\cdot}\tilde{r}_{0\cdot} + \frac{z_{1-\alpha/2}^2}{2}(\tilde{s}\tilde{s}_{0\cdot} + \tilde{r}\tilde{r}_{0\cdot} - 2\tilde{s}_{0\cdot}\tilde{r}_{0\cdot}) \pm z_{1-\alpha/2}\sqrt{\tilde{n}^2\tilde{s}_{0\cdot}\tilde{r}_{0\cdot}\left[\tilde{s}_{0\cdot} + \tilde{r}_{0\cdot} - \tilde{n}(1 - \tilde{S}e_1)\tilde{S}p_1\right] + \frac{z_{1-\alpha/2}^2}{4}(\tilde{s}\tilde{s}_{0\cdot} - \tilde{r}\tilde{r}_{0\cdot})^2}}{\tilde{r}_{0\cdot}\left[\tilde{n}\tilde{s}\tilde{S}p_1 - z_{1-\alpha/2}^2(\tilde{s} - \tilde{r}_{0\cdot})\right]}$$

where $\tilde{s}_{0\cdot} = s_{0\cdot} + 0.5$, $\tilde{r}_{0\cdot} = r_{0\cdot} + 0.5$, $\tilde{s} = s + 1$, $\tilde{r} = r + 1$, $\tilde{n} = n + 2$, $\tilde{S}e_1 = \tilde{s}_{1\cdot}/\tilde{s}$ and $\tilde{s}p_1 = \tilde{r}_{0\cdot}/\tilde{r}$.

Regarding NLR_1, if the lower limit of the confidence interval is less than $\tilde{s}_{0\cdot}/(\tilde{n} - \tilde{r}_{0\cdot})$ or greater than $N\hat{L}R_1$ then the lower limit is given by:

$$11. \quad \frac{\tilde{s}_{0\cdot}\tilde{S}p_1 + \frac{z_{1-\alpha/2}^2}{2} - z_{1-\alpha/2}\sqrt{\frac{z_{1-\alpha/2}^2}{4} + \tilde{s}_{0\cdot}(\tilde{S}p_1 + \tilde{S}e_1 - 1)}}{\tilde{s}\tilde{S}p_1^2 + z_{1-\alpha/2}^2},$$

Equally, if the upper limit is greater than $(\tilde{n} - \tilde{s}_{0\cdot})/\tilde{r}_{0\cdot}$ or less than $\widehat{PLR}_2$ then the upper limit is given by:

$$12. \quad \frac{\tilde{r}_{0\cdot}(1 - \tilde{S}e_1) + \frac{z_{1-\alpha/2}^2}{2} + z_{1-\alpha/2}\sqrt{\frac{z_{1-\alpha/2}^2}{4} + \tilde{r}_{0\cdot}(1 - \tilde{S}e_1 - \tilde{S}p_1)}}{\tilde{r}\tilde{S}p_1^2}.$$

Replacing $x_{0.}$ with $x_{.0}$ where x represents any of the fundamental values s, r or n and $\tilde{S}e_1$ and $\tilde{S}p_1$ with $\tilde{S}e_2$ and $\tilde{S}p_2$, respectively, will return the values for $N\hat{L}R_2$.

Using the package

The package consists of three main functions.

`compareR()`: This is the workhorse function of the package. It takes as its argument a data frame or matrix, which should be appropriately formatted (see 'Data preparation'). Internal functions then ensure data are correctly coded, before calculating output values according to the methodologies previously described. A range of optional parameters allow users to customise the output:

- `df` A data frame or matrix with at least 3 columns. Can be subset using `test1`, `test2` and `gold` arguments. If `test1`, `test2` and `gold` remain NULL then defaults to `test1 = df[,1]`, `test2 = df[,2]` and `gold = df[,3]`. Flexible coding of positive and negative results permitted.
- `test1` Either a vector of values for Test 1 (if `df` is NULL) or a string or integer value to be used for subsetting `df`.
- `test2` Either a vector of values for Test 2 (if `df` is NULL) or a string or integer value to be used for subsetting `df`.
- `gold` Either a vector of values for the `gold` standard test (if `df` is NULL) or a string or integer value to be used for subsetting `df`.
- `alpha` An alpha value, i.e. significance level. Defaults to 0.05.
- `margins` A Boolean value indicating whether the contingency tables should have margins containing summed totals of rows and columns. Defaults to FALSE.
- `multi_corr` Method for multiple comparisons. Uses `p.adjust.methods`. Defaults to "holm".
- `cc` A Boolean value indicating whether McNemar's test should be applied with continuity correction. Defaults to TRUE.
- `dp` Number of decimal places of output in summary tables. Defaults to 1.
- `sesp` A Boolean value indicating whether output should include sensitivity and specificity. Defaults to TRUE.
- `ppvnpv` A Boolean value indicating whether output should include positive and negative predictive values. Defaults to TRUE.
- `plrnlr` A Boolean value indicating whether output should include positive and negative likelihood ratios. Defaults to TRUE.
- `conf.int` A character string, either "contemporary" or "classic". Indicates whether the function should use the statistical methods described herein (contemporary, see Mathematical methods) or the exact binomial (classic) to calculate the confidence intervals.
- `test.names` A character vector of length two giving the names of the two different binary diagnostic tests. Defaults to c("Test 1", "Test 2").

The output from the `compareR()` function is a multilevel list object of class compareR. Users can access individual results using standard R indexing. The list structure is visually described in Figure 2. Alternatively, users can access the structure within R using a built-in data set with the command `str(compareR(cass))`.

`interpretR()`: The `interpretR()` function provides a means for clinicians to quickly understand the significance of their results, without having to manually dissect the multilevel list output from `compareR()`. By passing the `interpretR()` function the output from `compareR()` the user is provided with a readout in the console in plain English.

`summariseR()`: When a clinician is evaluating only one test, the `summariseR()` function will quickly calculate and display the descriptive statistics. Although this is not difficult to perform manually, the `summariseR()` function is fast and convenient, even with large datasets. Like `compareR()`, `summariseR()` allows flexible input, which can prevent researchers having to manually re-code their data. Users should note that unlike `compareR()`, `summariseR()` requires a data frame or matrix with two columns, evaluated test and gold standard test, as input.

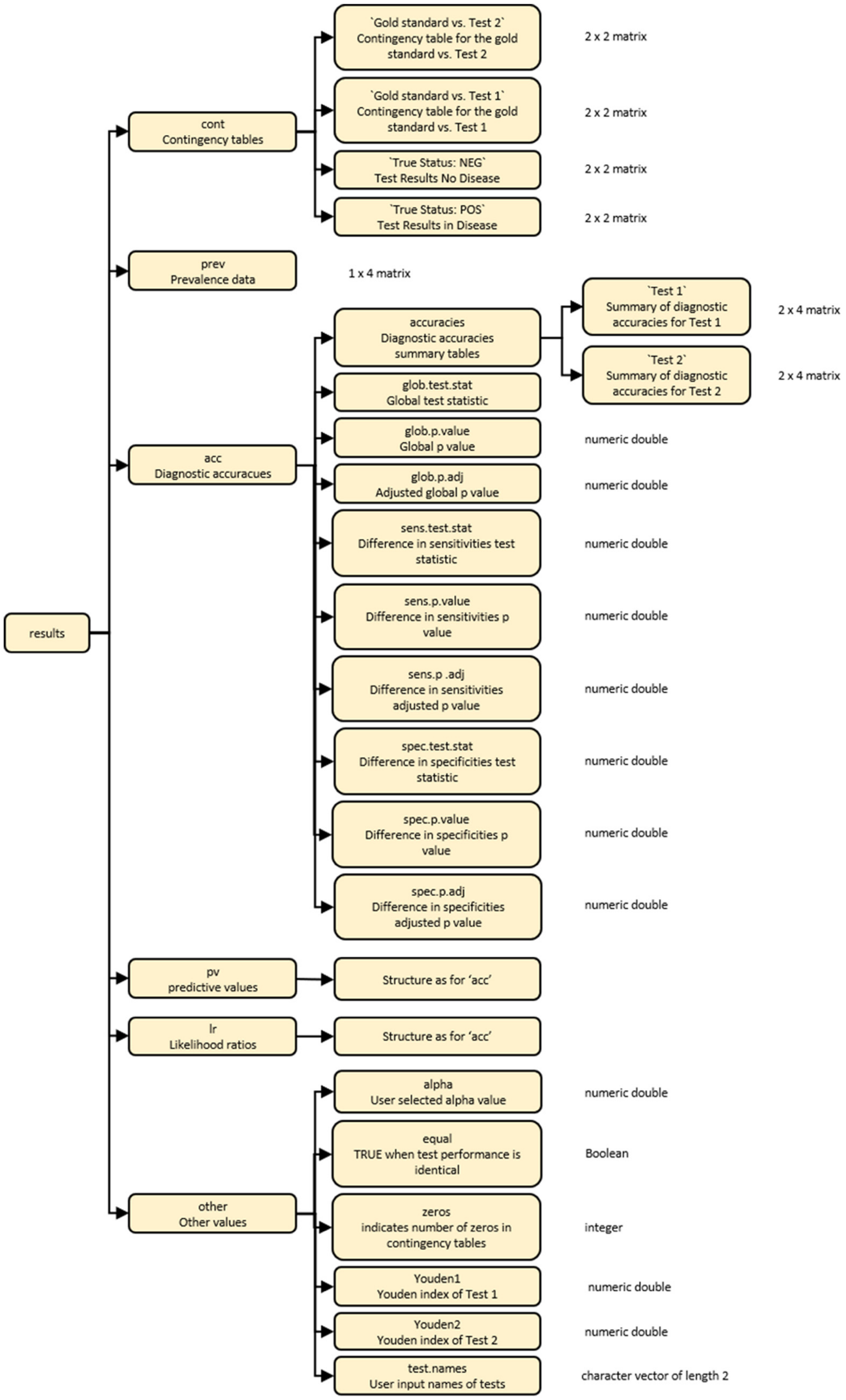

Figure 2. A diagrammatic representation of the multilevel list output from the compareR() function.

Because the outputs for both compareR and interpretR are verbose, testCompareR takes advantage of S3 methods within R's object-oriented programming system to provide the most useful results in more concise outputs. Users can get a summary() of the output, print() the most pertinent results as a table or plot() a simple visualisation of the diagnostic accuracies and predictive values.

Operation

At the time of writing, the testCompareR package can be run on any operating system that supports R version 4.3.0 or later.

Use cases

To demonstrate the use of the testCompareR package we will use the Coronary Artery Surgery Study (cass)[16] data set which is included with the package.

First, examining the data we see that the data frame contains three columns, exercise relating to an exercise stress test, cp relating to a history of chest pain (each used here as tests for coronary artery disease), and angio, which reports the outcome of the gold standard test – coronary angiography. Here, we can see that the data are already coded as zeros and ones.

```
data
(cass)
rbind(head(cass), tail(cass))
    exercise cp angio
1          1  1     1
2          1  1     1
3          1  1     1
4          1  1     1
5          1  1     1
6          1  1     1
866        0  0     0
867        0  0     0
868        0  0     0
869        0  0     0
870        0  0     0
871        0  0     0
```

To compare the two tests, pass the data to the compareR() function. This returns a multilevel list, as previously described. To avoid an unnecessary lengthy output in the example we have set the parameters ppvnpv and plrnlr to FALSE which allows us not to execute these tests.

```
results <- compareR(cass, ppvnpv = FALSE, plrnlr = FALSE)
results

$cont
$cont$`True Status: POS`
          Test 2
Test 1      Positive Negative
  Positive      473       29
  Negative       81       25

$cont$`True Status: NEG`
          Test 2
Test 1      Positive Negative
  Positive       22       46
  Negative       44      151

$prev
           Estimate  SE Lower CI Upper CI
Prevalence     69.8 1.6     66.7     72.8
```

```
$acc
$acc$accuracies
$acc$accuracies$`Test 1`
            Estimate  SE Lower CI Upper CI
Sensitivity     82.6 1.5     79.4     85.4
Specificity     74.1 2.7     68.6     79.1

$acc$accuracies$`Test 2`
            Estimate  SE Lower CI Upper CI
Sensitivity     91.1 1.2     88.6     93.1
Specificity     74.9 2.7     69.4     79.8

$acc$glob.test.stat
[1] 25.662

$acc$glob.p.value
[1] 2.676497e-06

$acc$glob.p.adj
[1] 2.676497e-06

$acc$sens.test.stat
[1] 23.64545

$acc$sens.p.value
[1] 0

$acc$sens.p.adj
[1] 0

$acc$spec.test.stat
[1] 0.01111111

$acc$spec.p.value
[1] 0.9911348

$acc$spec.p.adj
[1] 1

$other
$other$alpha
[1] 0.05

$other$equal
[1] FALSE

$other$zeros
[1] 0

$other$Youden1
[1] 0.5671028

$other$Youden2
[1] 0.6602336

$other$test.names
[1] "Test 1" "Test 2"

attr(,"class")
[1] "compareR"
```

Values in this list can be accessed via standard indexing.

```
results$acc$accuracies # returns matrices summarising diagnostic accuracies
$`Test 1`
            Estimate  SE Lower CI Upper CI
Sensitivity     82.6 1.5     79.4     85.4
Specificity     74.1 2.7     68.6     79.1

$`Test 2`

            Estimate  SE Lower CI Upper CI
Sensitivity     91.1 1.2     88.6     93.1
Specificity     74.9 2.7     69.4     79.8
```

Finally, if the user prefers to see an interpretation of the output in plain English, including highlighted values where results are significant, they can pass the output of `compareR()` to `interpretR()`.

```
interpretR(results)
----------------------------------------------------------------------------
----
CONTINGENCY TABLES
----------------------------------------------------------------------------
----

True Status - POSITIVE
          Test 2
Test 1     Positive Negative
  Positive      473       29
  Negative       81       25

True Status - NEGATIVE
          Test 2
Test 1     Positive Negative
  Positive       22       46
  Negative       44      151

----------------------------------------------------------------------------
----
PREVALENCE (%)
----------------------------------------------------------------------------
----

           Estimate  SE Lower CI Upper CI
Prevalence     69.8 1.6     66.7     72.8

----------------------------------------------------------------------------
----
DIAGNOSTIC ACCURACIES
----------------------------------------------------------------------------
----

 Test 1 (%)
            Estimate  SE Lower CI Upper CI
Sensitivity     82.6 1.5     79.4     85.4
Specificity     74.1 2.7     68.6     79.1
```

```
 Test 2 (%)

            Estimate  SE Lower CI Upper CI
Sensitivity     91.1 1.2     88.6     93.1
Specificity     74.9 2.7     69.4     79.8

Global Null Hypothesis: Se1 = Se2 & Sp1 = Sp2
Test statistic:  25.662  Adjusted p value:  2.676497e-06 ***SIGNIFICANT***

Investigating cause(s) of significance

Null Hypothesis 1: Se1 = Se2
Test statistic:  23.64545  Adjusted p value:  0 ***SIGNIFICANT***

Null Hypothesis 2: Sp1 = Sp2
Test statistic:  0.01111111  Adjusted p value:  1
```

`testCompareR` is elegant in its simplicity. Several parameters permit customisation of the output, but they are not elaborated here as they are not essential to understand the workings of the package (the interested reader is referred to the `testCompareR` package documentation files). Further details can be found within the package vignette, which contains examples for all modifiable parameters.

Results

Evaluation

We calculated the results for the Coronary Artery Surgery Study (`cass`) dataset using `testCompareR`, `DTComPair` and `compbdt`. This dataset looks at exercise stress testing and history of chest pain as two tests for coronary artery disease as determined by coronary angiography (the gold standard)[16]. It has become a standard for testing in statistical research regarding test metrics. The results are shown in Table 5.

Both `testCompareR` and `DTComPair` both achieve similar results. Given that the mathematical basis of `testCompareR` and `compbdt` is the same, we see almost identical results. However, the test statistics for the individual hypothesis tests of the diagnostic accuracies (sensitivity and specificity) are distributed approximately according to the chi-squared distribution, which is reflected in the `testCompareR` package. The original `compbdt` program mistakenly treats these test statistics as if they were distributed according to the normal distribution which can lead to differences between results (see e.g. results for the diagnostic accuracy of Test 1 in Table 5).

Performance evaluation

To evaluate the performance of the package we used the `microbenchmark` package[17] to repeatedly compute the test metrics, calculate confidence intervals and perform inferential tests on the `cass` dataset using `testCompareR`, `DTComPair` and `compbdt`. Each set of calculations was repeated 100 times and the time elapsed for each test was recorded. The results are shown in Table 6 and Figure 3A.

A certain amount of pre-processing was required in order that the data conformed to the requirements of each package or program. The code describing this pre-processing is included with this paper.

Our results demonstrate that the `compareR()` function returns the results considerably quicker than either `DTComPair` or `compbdt`. This performance advantage is maintained even when `compareR()` is wrapped by the `interpretR()` function. Further testing of the individual functions from `DTComPair` and the internal functions of `testCompareR` demonstrated that the cause of the difference between the two packages is the method for comparing the likelihood ratios, shown in Figure 3B. The method used by `DTComPair` is based on logistical regression, whereas the method used by `testCompareR` is based on an approximation of the score statistic, which is simpler to compute requiring only solving of a second-degree equation. Simulation to estimate the power and type 1 error rate for this approximation across a large range of scenarios found that it performs well in most reasonable use cases[15].

Table 5. Outputted values for each of the test metrics by package/program. DTComPair we were unable to retrieve prevalence from DTComPair's standard functions. For likelihood ratios testCompareR and compbdt report SE, whereas DTComPair reports SE.log, therefore they are not directly comparable. Se - sensitivity, SE - standard error, LCI - lower confidence interval, UCI - upper confidence interval, p - p value, Sp - specificity, PPV - positive predictive value, NPV - negative predictive value, PLR - positive likelihood ratio, NLR - negative likelihood ratio.

Disease prevalence										
	Prevalence	**SE**	**LCI**	**UCI**						
testCompareR	69.8	1.6	66.7	72.8						
compbdt	69.8	1.6	66.7	72.8						
Diagnostic accuracies: Test 1										
	Se	**SE**	**LCI**	**UCI**	**p**	**Sp**	**SE**	**LCI**	**UCI**	**p**
testCompareR	82.6	1.5	79.4	85.4	<0.0001	74.1	2.7	68.6	79.1	0.92
DTComPair	82.6	1.5	79.6	85.6	<0.0001	74.1	2.7	68.9	79.4	0.83
compbdt	82.6	1.5	79.4	85.4	<0.0001	74.1	2.7	68.6	79.1	0.99
Diagnostic accuracies: Test 2										
	Se	**SE**	**LCI**	**UCI**	**p**	**Sp**	**SE**	**LCI**	**UCI**	**p**
testCompareR	91.1	1.2	88.6	93.2		74.9	2.7	69.4	79.8	
DTComPair	91.1	1.2	88.9	93.2		74.9	2.7	69.7	80.1	
compbdt	91.1	1.2	88.6	93.2		74.9	2.7	69.4	79.8	
Predictive values: Test 1										
	PPV	**SE**	**LCI**	**UCI**	**p**	**NPV**	**SE**	**LCI**	**UCI**	**p**
testCompareR	88.1	1.4	85.2	90.5	0.37	64.8	2.8	59.3	70.0	<0.0001
DTComPair	88.1	1.4	85.4	90.7	0.37	64.8	2.8	59.4	70.2	<0.0001
compbdt	88.1	1.4	85.2	90.5	0.37	64.8	2.8	59.3	70.0	<0.0001
Predictive values: Test 2										
	PPV	**SE**	**LCI**	**UCI**	**p**	**NPV**	**SE**	**LCI**	**UCI**	**p**
testCompareR	89.4	1.2	86.7	91.6		78.5	2.6	73.0	83.2	
DTComPair	89.4	1.2	86.9	91.8		78.5	2.6	73.4	83.6	
compbdt	89.4	1.2	86.7	91.6		78.5	2.6	73.0	83.2	
Likelihood ratios: Test 1										
	PLR		**LCI**	**UCI**	**p**	**NLR**		**LCI**	**UCI**	**p**
testCompareR	3.2		2.6	4.0	0.37	0.23		0.20	0.28	<0.0001
DTComPair	3.2		2.6	3.9	0.37	0.23		0.20	0.28	<0.0001
compbdt	3.2		2.6	4.0	0.37	0.23		0.20	0.28	<0.0001
Likelihood ratios: Test 2										
	PLR		**LCI**	**UCI**	**p**	**NLR**		**LCI**	**UCI**	**p**
testCompareR	3.6		3.0	4.5		0.12		0.09	0.15	
DTComPair	3.6		2.9	4.5		0.12		0.09	0.15	
compbdt	3.6		3.0	4.5		0.12		0.09	0.15	

Table 6. Time to compute descriptive and inferential statistics. Times computed using the `cass` dataset for individual packages/programs using a Windows 10 x64 laptop with 16GB RAM and an 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz processor. `testCompareR` was run in R 4.3.0 via Rstudio 2023.12.1+402.

	Mean (ms)	SD (ms)
compareR	14.1	7.8
interpretR	20.5	11.4
DTComPair	114.2	38.1
compbdt	3,578.4	1,752.3

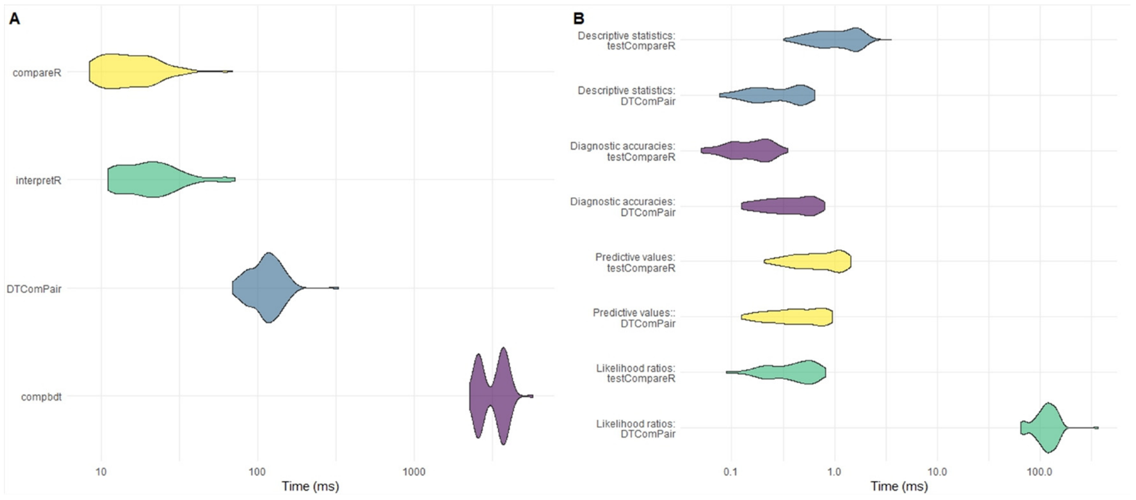

Figure 3. A) Comparison of computational time for testCompareR, DTComPair and compbdt. Each package was run 100 times. **B**) Comparison of testCompareR and DTComPair, by function. testCompareR compares the likelihood ratios more quickly than DTComPair because it uses a method based on an approximation of the score statistic, which requires only solving of a second degree equation. The DTComPair method is based upon logistical regression. Each function was run 100 times.

To evaluate how the size of the data set affects the computational efficiency of `testCompareR` and `DTComPair` we again used the `microbenchmark` package to time 25 evaluations of the functions for simulated data sets containing between 20000 and 100000 participants in intervals of 20000. These data sets are significantly larger than data sets that one would expect to find in diagnostic accuracy studies and therefore testing beyond this range was deemed unnecessary. `compareR()`, `interpretR()` and `DTComPair` all scaled approximately linearly as the data set size increased (see Figure 4). When simultaneously evaluating all the built-in testing functions on very large datasets, testCompareR still performs in under 1 second. The major contributor to slower performance in DTComPair is comparison of likelihood ratios, though this is likely only of practical importance for very large datasets.

Discussion

Despite the common use of binary diagnostic tests in medicine only one package, `DTComPair`, and one open-source program, `compbdt`, provide methods to compare the test metrics between two binary diagnostic tests using paired data[5,6]. This package requires the user to be computationally literate, as several function calls are necessary to extract the outputs that would normally be published when comparing the performance

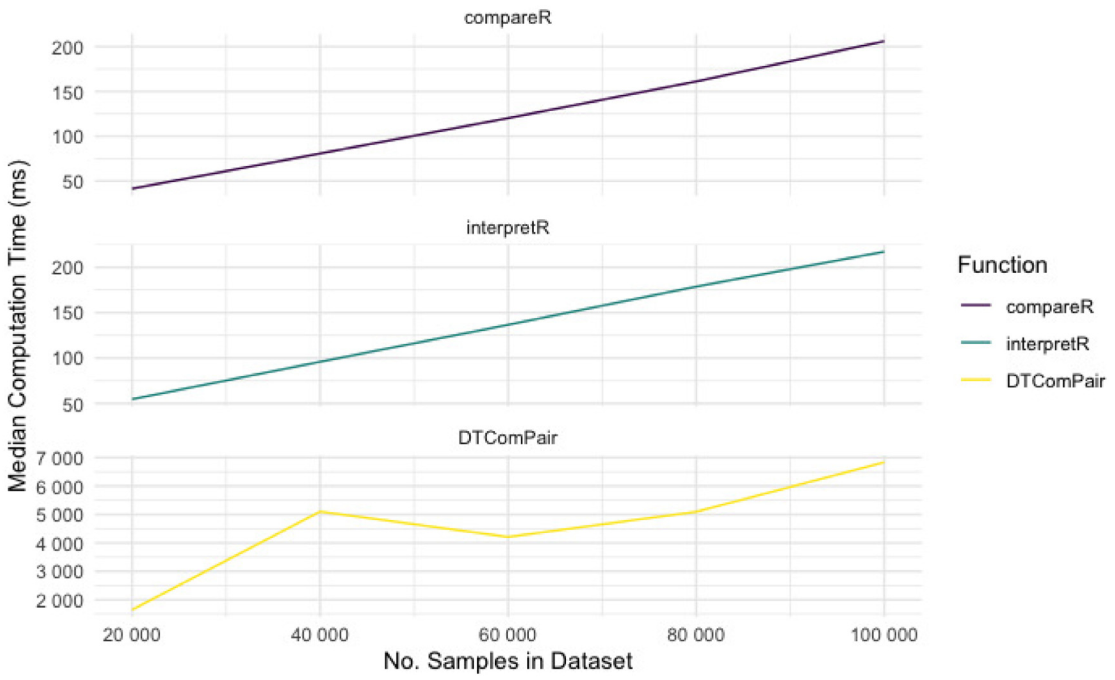

Figure 4. Variation in computation time by data set size, by package / function.

of two tests. Specifically, the data must be pre-processed adequately and fed to the package as an object of class `tab.paired`, which is specific to the package. It would be more intuitive if `DTComPair` accepted as its input a more general input data class, such as a data frame, matrix or a tibble. Additionally, though the package implements well-established traditional methods, the evidence suggests that newer methods provide better coverage in the case of confidence intervals and better asymptotic performance in the case of hypothesis tests[10–15]. Here we have shown with an example dataset that the newer methods for comparing the likelihood ratios between two tests achieve comparable results, but are more computationally efficient.

The `compbdt` program requires the user to find the program in the statistical literature, copy or download the code, load the function, preprocess the data and then run the function. Customisations to the output require the user to update the code. The output is in the form of a lengthy console readout, which can make downstream analyses challenging. By re-structuring the internal mechanisms, we have increased computational speed while providing additional features: `testCompareR` can be installed directly from CRAN; accepts a dataframe as an argument; requires minimal preprocessing; and users can customise the output through a range of well-documented optional arguments. The user can choose whether to receive their output in list form, allowing them to access individual elements for downstream analysis via indexing, or as a plain English summary, facilitating rapid interpretation of the results.

`testCompareR` adds to the arsenal of tools for researchers who wish to rapidly develop and evaluate diagnostic tests. By minimising the number of steps required for analysis, `testCompareR` frees up valuable time for laboratory and clinical research.

Ethics and consent
This research involved neither human nor animal participants so no consent or ethical approvals were required.

Data and software availability
The data described in this paper is available within the package. The data was originally presented by Weiner *et al.* as part of the Coronary Artery Surgery Study (CASS)[16]. This study used data from a national registry set-up and maintained by the Division of Heart & Vascular Disease at the Heart, Lung and Blood Institute of the National Institute of Health (USA).

Source code available from: https://www.github.com/Kajlinko/testCompareR

Archived software available from: https://zenodo.org/doi/10.5281/zenodo.11488420[18].

License: GPL-3.0+

Author contributions

Kyle J. Wilson: conceptualisation, methodology, software, validation, writing - original draft preparation, writing – reviewing and editing.

José A. Roldán-Nofuentes: methodology, software, validation, writing – reviewing and editing.

Marc Y. R. Henrion: supervision, validation, writing - reviewing and editing.

Acknowledgements

We would like to acknowledge the support of Dr Nicholas Beare. Additionally, we acknowledge the contribution of Alice Liomba, whose research question led to the development of this software.

References

1. Valanis BG, Perlman CS: **Home pregnancy testing kits: prevalence of use, false-negative rates, and compliance with instructions.** *Am J Public Health.* 1982; **72**(9): 1034–6.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**
2. Crawford F, Andras A, Welch K, *et al.*: **D-dimer test for excluding the diagnosis of pulmonary embolism.** *Cochrane Database Syst Rev.* 2016; **2016**(8): CD010864.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**
3. Budd J, Miller BS, Weckman NE, *et al.*: **Lateral flow test engineering and lessons learned from COVID-19.** *Nat Rev Bioeng.* 2023; **1**(1): 13–31.
Publisher Full Text
4. R Core Team: **R: A Language and Environment for Statistical Computing.** Vienna, Austria: R Foundation for Statistical Computing; 2023.
Reference Source
5. Stock C, Hielscher T, Discacciati A: **DTComPair: comparison of binary diagnostic tests in a paired study design.** R package, version 1.2.0. 2023.
Reference Source
6. Roldán-Nofuentes JA: **Compbdt: an R program to compare two binary diagnostic tests subject to a paired design.** *BMC Med Res Methodol.* 2020; **20**(1): 143.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**
7. Altman DG, Bland JM: **Diagnostic tests. 1: sensitivity and specificity.** *BMJ.* 1994; **308**(6943): 1552.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**
8. Altman DG, Bland JM: **Diagnostic tests 2: predictive values.** *BMJ.* 1994; **309**(6947): 102.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**
9. Deeks JJ, Altman DG: **Diagnostic tests 4: likelihood ratios.** *BMJ.* 2004; **329**(7458): 168–9.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**
10. Yu W, Guo X, Xu W: **An improved score interval with a modified midpoint for a binomial proportion.** *J Stat Comput Simul.* 2014; **84**(5): 1022–38.
Publisher Full Text
11. Martín Andrés A, Álvarez Hernández M: **Two-tailed approximate confidence intervals for the ratio of proportions.** *Stat Comput.* 2014; **24**(1): 65–75.
Publisher Full Text
12. Roldán-Nofuentes JA, Sidaty-Regad SB: **Recommended methods to compare the accuracy of two binary diagnostic tests subject to a paired design.** *J Stat Comput Simul.* 2019; **89**(14): 2621–44.
Publisher Full Text
13. Roldán Nofuentes JA, Luna Del Castillo JDD, Montero Alonso MÁ: **Global hypothesis test to simultaneously compare the predictive values of two binary diagnostic tests.** *Comput Stat Data Anal.* 2012; **56**(5): 1161–73.
Publisher Full Text
14. Kosinski AS: **A weighted generalized score statistic for comparison of predictive values of diagnostic tests.** *Stat Med.* 2013; **32**(6): 964–77.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**
15. Nofuentes JAR, del Castillo J de DL: **Comparison of the likelihood ratios of two binary diagnostic tests in paired designs.** *Stat Med.* 2007; **26**(22): 4179–201.
PubMed Abstract | **Publisher Full Text**
16. Weiner DA, Ryan TJ, McCabe CH, *et al.*: **Exercise stress testing. Correlations among history of angina, ST-segment response and prevalence of coronary-artery disease in the Coronary Artery Surgery Study (CASS).** *N Engl J Med.* 1979; **301**(5): 230–5.
PubMed Abstract | **Publisher Full Text**
17. Mersmann O: **microbenchmark: accurate timing functions**.
Reference Source
18. Wilson KJ: **Kajlinko/testCompareR: testCompareR v1.0.3 (v1.0.3).** *Zenodo*. [Software]. 2024.
http://www.doi.org/10.5281/zenodo.11488433

Open Peer Review

Current Peer Review Status: ✓ ? ?

Version 3

Reviewer Report 04 November 2024

https://doi.org/10.21956/wellcomeopenres.25667.r105033

?

Haydar Demirhan
[1] RMIT University, Melbourne, Victoria, Australia
[2] RMIT University, Melbourne, Victoria, Australia

This study introduces an R package called testCompareR to compare the test metrics using two binary diagnostic tests. The features of the packages are discussed, and the statistical methods behind the proposed package are given clearly. Some examples are given to demonstrate the usage of the main functions of the package. My comments and questions are as follows:

1. The meaning of "robust statistical package" is unclear in the abstract.

2. In the abstract - background, it is mentioned that "...performing inferential statistics to compare commonly used test metrics for two diagnostic tests have not yet been implemented in a robust statistical package." However, in the abstract - methods, it is mentioned that another alternative is available.

3. The features of testCompareR are listed in the introduction. However, how many of these features are also covered in DTComPair is unclear. A comparison table could be given to make the contribution of testCompareR clearer. In its current form, it is hard to follow the contribution and necessity of the proposed package.

4. The contributions of this study to the existing literature need to be articulated clearly in the introduction.

5. I guess the information given under the title "Mathematical descriptions" is more of "Statistical descriptions."

6. It is hard to read Figure 2. The authors may consider changing the orientation of the figure to landscape.

7. Does the function interpretR() have any arguments? From the help files of the package, it seems

that there are no arguments. Thus, why not include another Boolean argument in compareR() to get the results interpreted?

8. What is the reason for giving the run time of compareR() and interpret() separately in Table 5?

9. How does the run time change when the size of the dataset is increased?

10. What factors impact the computational efficiency of the benchmarked packages?

Is the rationale for developing the new software tool clearly explained?
Partly

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Partly

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Partly

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Statistical software, Bayesian modelling, bioinformatics, categorical data analysis.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Reviewer Report 23 October 2024

https://doi.org/10.21956/wellcomeopenres.25667.r106828

✓ **Dennis Robert**

[1] Director of Clinical Research (AI), Qure.ai, Bangalore, India
[2] Director of Clinical Research (AI), Qure.ai, Bangalore, India

Thank you for incorporating my suggestions. The revised version of the article is much improved version. I have no further comments.

Is the rationale for developing the new software tool clearly explained?
Partly

Is the description of the software tool technically sound?
Partly

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Partly

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Partly

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Partly

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Statistical Methods in Diagnostic Medicine, Medical AI evaluation, R Statistical Computing

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Version 1

Reviewer Report 22 September 2024

https://doi.org/10.21956/wellcomeopenres.24690.r97069

? **Aaron Caldwell**
[1] University of Arkansas for Medical Sciences, Arkansas, USA

[2] University of Arkansas for Medical Sciences, Arkansas, USA
[3] University of Arkansas for Medical Sciences, Arkansas, USA
[4] University of Arkansas for Medical Sciences, Arkansas, USA

The authors have presented an interesting R package for comparing two diagnostic tests against a gold standard (or rather ground truth). I have no serious issues with the statistical methods, however I have significant concerns on the usability of the software in its current form. Another review has already detailed many other points. My review will detail a few issues that should be addressed to help the end user of this package and aide in the interpretation.

Major Comments:

1. I think it would be preferable for users to explicitly state which columns correspond to test1, test2 and "gold". As it currently stands, users are at the mercy of the package and have to follow a strict approach and provide a dataset with exactly 3 columns and in a particular order as well. I think it would be much more user-friendly for users to explicitly declare the columns which correspond to which test. So compareR should have three arguments: "test1", "test2", and "gold". These could be set to NULL as default and the function continue to be used like it is now. Otherwise, these could be explicitly declared and those three columns selected (even from a data frame of many columns). The code in the function could look as follows:

```
compareR <- function(df,
test1 = NULL, test2=NULL, gold = NULL,
alpha = 0.05, margins = FALSE, multi_corr = "holm",
                    cc = TRUE, dp = 1,
                    sesp = TRUE, ppvnpv = TRUE, plrnlr = TRUE,
                    conf.int = "contemporary",
                    test.names = c("Test 1", "Test 2"), ...) {
if( !is.null(test1) | !is.null(test2) | !is.null(gold)){
if( is.null(test1) | is.null(test2) | is.null(gold){
stop("At least one column named provided. Must provide other column names.")
}
df = df[c(test1,test2,gold)]
}
  df <- recoder(df)
```

2. The output of the functions is extremely verbose (even when passed to interpretR), and the default output for the class "compareR" is just a list. It would be useful to create & use S3 or S4 methods to simplify the output of results for users. For example, a "print" method could provide a very simple, not verbose, description of the results with arguments within the print method to allow for different items to be output to the console. A "plot" method could provide simple visualization. Other methods could be utilized to provide other types of output (e.g., formatted tables). I think packages from the easystats group (e.g., effectsize or performance) would be good examples to get an idea of what would be better practice. I think even a report function could be used to provide a simple summary/interpretation of the results in text (see the report R package as an example).

Is the rationale for developing the new software tool clearly explained?

Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Partly

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Partly

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Biostatistics

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Author Response 01 Oct 2024

Kyle Wilson

Thank you for the detailed feedback on our manuscript. It is clear from your report that you have taken the time to review not only the paper, but the code and package documentation. The effort is greatly appreciated; your suggestions have been integrated and the package is better for it. Beneath is a point-by-point response to your comments:

1. This is an excellent point and was a definite weakness in the package. I have implemented this suggestion via a function, which gives us some flexibility to adjust testCompareR further in the future. We also felt that the users may wish to provide the data as atomic vectors and leave the df argument as NULL. This has been implemented in the new function to validate the data. It is still possible for users to supply a data frame with no indexing for the columns, but in this case testCompareR will issue a warning so that users are aware that it may introduce error.

2. Another excellent suggestion. Most users probably don’t need to access all the different levels of the list output. Some may choose to access unadjusted p values, which they still can do through usual indexing. For users who prefer a less verbose output I have provided the following three S3 methods:

summary() – this provides a title, the names of the individual tests (if given), and the contingency tables for each test against the gold standard, as well as a table of the results.

It is informative and not verbose. print() – this provides a table of results. This is the bare minimum information required to review the analysis. plot() – this provides a very simple box and whisker plot for diagnostic accuracies and predictive values. We decided not to include likelihood ratios on this plot because they are not confined by the 0-1 scale and therefore compress the confidence intervals for the other metrics. Users who are particularly interested in likelihood ratios are probably more statistically savvy and therefore are likely able to plot them themselves.   Many thanks, once again, for taking a deep-dive into testCompareR. The valuable feedback has improved the manuscript and the package. If you have further comments, they will be considered and integrated. Otherwise, we hope you are satisfied with the updated manuscript.

Competing Interests: No competing interests were disclosed.

Reviewer Report 11 September 2024

https://doi.org/10.21956/wellcomeopenres.24690.r97075

? **Dennis Robert**

[1] Director of Clinical Research (AI), Qure.ai, Bangalore, India
[2] Director of Clinical Research (AI), Qure.ai, Bangalore, India
[3] Director of Clinical Research (AI), Qure.ai, Bangalore, India
[4] Director of Clinical Research (AI), Qure.ai, Bangalore, India

SUMMARY:

I have read the article with great interest. The authors developed an R package titled *testCompareR* which can enable researchers to perform comparison of diagnostic accuracies of two binary diagnostic tests using paired data. Such comparisons are commonly conducted and this means that this package could be potentially very useful to many researchers. There are two similar R packages already available - *DTComPair* and *compbdt,* and the authors presented a detailed comparison of testCompareR with these already available existing two packages. testCompareR uses more statistically robust and up-to-date algorithms and has a very intuitive user interface that requires limited pre-processing steps. The article is reported in a mostly transparent manner. I congratulate the authors for developing this package as well as for reporting the details in a manuscript. However, I think the article can be improved by adding some more clarifications. Please find my list of comments below.

COMMENTS:

1. In the ABSTRACT, the authors quote “*The output and efficiency of testCompareR is compared to the only other available package which performs this function, DTComPair, using a motivating example.”* This has to be modified because the comparison was not only done with

DTComPair, but also with *compbdt*. While the latter package may not be available in CRAN, it is still available as an open-source code for anyone to use.

2. In INTRODUCTION, the authors quote: *"A newer program, compbdt, has also been published in an open-access journal. This program uses the most up-to-date statistical methods".* As I understand, the two major motivations for the authors to develop the *testCompareR* R package is that 1) the package includes the so called "up-to-date statistical methods" and 2) *testCompareR* requires less cumbersome data pre-processing steps than *DTComPair* and *compbdt*. While it is true that the authors clearly documented the motivation, the details on what exactly are these *"up-to-date statistical methods",* is missing. *DTComPair* uses McNemar's test to compare sensitivity and specificity of two binary diagnostic tests in a paired study design. *testCompareR* also seems to be using the same test. Are the "up-to-date statistical methods" referring only to confidence interval construction based on Yu et.al. or do they mean something more?. This could better be explicitly clarified in the METHODS section.

3. In Table 1, there seems to be a typo. False negative (FN) and False positive (FP) should be inter-changed so that the table reflects the correct definitions of FP (gold standard is negative, but test is positive) and FN (gold standard is positive, but test is negative).

4. In ***"Mathematical descriptions",*** please number the equations so that it can be easily referenced.

5. In ***"Mathematical descriptions",*** some of the terms in the equations are not defined. For example, what is "s", "r", "Se_i", etc. I could not find these terms in Table 3 too. If I refer to the article by Yu et. al., it looks like "s" represents the number of diseased (positive) cases and "r" represents the number of non-diseased (negative) cases (the ratio of "s" and "s+r" will thus be the estimator of prevalence), but this should be documented in this article too to improve clarity. Please define all the terms clearly.

6. The section "Use cases" should be placed before the "Results"/"Evaluation" section. This is because the "Use cases" illustrate how the functions in the testCompareR works and it should follow the *"Using the package"* section. This will allow the users to first understand how the programming works with the functions in the package. After "Use cases", the "RESULTS" can show the "Evaluation" results comparing with other packages.

7. Table 4 contains results with 2 decimal point precision. However, the numbers in the output under "Use cases" shows the results with single decimal point precision because you have used the default value of 1 for the argument *dp* in the function *compareR*. Please make the decimal point precision consistent between all text and tables whenever applicable.

8. Can the p values be reported in Table 4 like $p < .0001$, $p < .001$, or if p is $\geq .001$, report exact p with 2 decimal point precision? The exception to this rule is when rounding p from 3 digits to 2 digits would result in p appearing non-significant (such as p=.047).

9. It is unclear how the standard error and confidence intervals for the point estimates of diagnostic accuracy like sensitivity or specificity are obtained from DTComPair package. I assume that the function in DTComPair that is comparable with the "compare" function of

"testCompareR" is "sesp.mcnemar". However, sesp.mcnemar returns only the point estimates, test statistics, difference between point estimates and p value. The confidence interval for the difference between the point estimates are retrievable from using "sesp.diff.ci" function. But as I mentioned, it returns the CI for the difference in point estimates and not the point estimates per se. Can the authors include more details so as to make enable reproducible results? I have not tried to do this using the "compbdt" package. In fact, a fully reproducible R script that can verify the results from all three packages may be added as an Appendix/Supplement to faciliate this. I believe this code must already be available with the authors. This addition would tremendously improve the quality of the manuscript.

10. Please add a line of code "data(cass)" above the line "rbind(head(cass), tail(cass))" so that the data.frame appears in an R IDE like in Rstudio environment. This enables the users to view the data.frame easily by simply clicking the object in environment. As you have mentioned, the users of this package may be less code-savvy (like clinicians).

11. testCompareR's compare function has an argument named "conf.int" which seems to be a fundamental argument as per the R package help document but is missing in the paper. Please include this under the list of "a range of optional parameters allow users to customise the output".

12. The three arguments – sesp, ppvnpv, plrnlr, seems to be somewhat related. For example, if all three are FALSE, then the function will return an error. It seems at least one of them has to declared TRUE. While the error output in R console says, "No tests selected", I think the message can be clearer; something like "No tests selected. At least one among sesp, ppvnpv and plrnlr should be declared TRUE".

13. In DISCUSSION, the authors quote *"Despite the common use of binary diagnostic tests in medicine only one package, DTComPair, provides methods to compare the test metrics between two binary diagnostic tests using paired data".* What about compbdt?

14. In DISCUSSION, the authors quote *"This package requires the user to be computationally literate, as several function calls are necessary to extract the outputs that would normally be published when comparing the performance of two tests".* I agree with the authors. It is not very intuitive to use DTComPair functions, even for experienced R users. DTComPair actually requires input data to be fed in a custom (specific to DTComPair package) format of an object of class tab.paired. The ideal way would be to feed more popular and general input data class like data.frame which is what makes testCompareR very intuitive to use. Perhaps this can also be noted down in the DISCUSSION.

15. In DISCUSSION, the authors quote "*By re-structuring the internal mechanisms, we have dramatically increased computational speed while providing additional features".* While the authors have shown that the computational efficiency of testCompareR compared to DTComPair and compbdt packages in RESULTS, I don't think the use of word "dramatically" is very appropriate here. Most of these function executions will only take maximum few seconds only in most scenarios, regardless of the package. I suggest toning down the language. In fact, in Figure 2B, there is considerable overlap between the distribution of computational time of testCompareR and DTComPair, except for the computation of

likelihood ratios. Moreover, the main added values I see with testCompareR is its intuitive user interface (this is great, by the way!) and usage of up-to-date statistical algorithms for construction of confidence intervals with better coverage. The reduction in computational time, although true, is perhaps not that important/impactful.

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Partly

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Statistical Methods in Diagnostic Medicine, Medical AI evaluation, Real-World-Data and R Programming

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Author Response 18 Sep 2024
Kyle Wilson

Thank you for the detailed feedback on our manuscript. It is clear from your report that you have taken the time to review not only the paper, but the code and package documentation. The effort is greatly appreciated; your suggestions have been integrated and the package is better for it. Beneath is a point-by-point response to your comments:

1. A line has been added to the abstract to indicate that testCompareR is evaluated against the only available package, DTComPair, and the open-source program, compbdt.
2. The statistical methods are as follows:
    - Yu et al. interval for confidence intervals of diagnostic accuracies and PVs
    - Approximation of the score method for confidence intervals for LRs (as described by Martín Andrés and Álvarez Hernández in their review - ref 11)

- Testing the difference in diagnostic accuracies (Se and Sp) uses a complex algorithm, whereby the method is adjusted based upon prevalence and total number of particpants / samples. It seems that the ‘wall-of-text’ explaining this method is not clear in and of itself, so we have supplemented this with a diagrammatic decision tree, which better illustrates the process.
- Testing the difference in predictive values uses a global test based on the Wald test statistic. The cause of the significance is investigated using Kosinski’s weighted generalised score statistic.
- Testing the difference in likelihood ratios uses the Wald test considering the natural logarithm of the ratios of positive and negative likelihood ratios.

3. The typo is well-noted and has been corrected.
4. Thanks for this advice. The equations have been numbered.
5. The new version manuscript has been updated accordingly so that all terms in the equations are explicitly defined.
6. This section has now been moved as you suggest.
7. The table has been updated to maintain consistency throughout the manuscript. The exception here is the values for NLR. As these ratios are often very small, 2 significant figures were maintained. While this could be updated again at your request, it was felt that this retains important information without sacrificing consistency overall. It was also noted that somewhere in the editing process the values for PV1 were copied into PV2. They were identical. This has been updated while addressing Point 9.
8. The p values have been updated as you suggest. This makes the message clearer. Thanks.
9. You are correct that the benchmarking script already exists. This has been uploaded to the Github version of the repository and excluded from R builds (so it will not be on CRAN, in accordance with their formatting guidance) in the folder benchmarking. This Github repository has then been archived to Zenodo to maintain it for posterity. The script was cleaned up and transferred to a Quarto document, to make it more readable for interested users. The code was then re-run on a 16GB MacBook Pro with M2 Pro Silicon Chip running Ventura 13.5.1. As there were not major changes to the figures, they have not been updated, but it is nice to have seen (and to be able to demonstrate) the package functioning well on another system, also.
10. This line of code has been added.
11. This has now been updated. The conf.int argument allows users to select the Yu et al. confidence interval (“contemporary”) or the exact binomial confidence interval (“classic”) when calculating CIs for binomial proportions.
12. This is a good suggestion. This has been integrated into the code and it now fails more elegantly.
13. The discussion has been updated to reflect the availability of compbdt. We have been explicit that this is not a package, but a program.
14. Thanks again for this insightful remark. The manuscript has been updated to reflect the specifics of the data input requirements of DTComPair.
15. While the speed with which testCompareR performs is several orders of magnitude faster than compbdt, and one order of magnitude faster than DTComPair, it is very unlikely that an average user will appreciate the difference as meaningful. The 100ms

processing time for DTComPair is practically instantaneous. The word dramatically has been removed.

Many thanks, once again, for taking a deep-dive into testCompareR. The valuable feedback has improved the manuscript and the package. If you have further comments, they will be considered and integrated. Otherwise, we hope you are satisfied with the updated manuscript.

Competing Interests: No competing interests were disclosed.

Reviewer Response 12 Oct 2024

Dennis Robert

I have no further comments as all my previous comments have been adequately addressed by the author (s). The version 2 of the manuscript is a much improved version. Great work and I am sure the testCompareR will benefit a lot of researchers worldwide.

Competing Interests: No competing interests were disclosed.