SOFTWARE TOOL ARTICLE

testCompareR: an R package to compare two binary diagnostic tests using paired data [version 1; peer review: awaiting peer review]

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

v1 **First published:** 02 Jul 2024, **9**:351
https://doi.org/10.12688/wellcomeopenres.22411.1
Latest published: 02 Jul 2024, **9**:351
https://doi.org/10.12688/wellcomeopenres.22411.1

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

Open Peer Review

Approval Status *AWAITING PEER REVIEW*

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

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

This article is included in the Malawi-Liverpool Wellcome Trust Clinical Research Programme gateway.

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.

Copyright: © 2024 Wilson KJ *et al*. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

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 1; peer review: awaiting peer review]** Wellcome Open Research 2024, **9**:351 https://doi.org/10.12688/wellcomeopenres.22411.1

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

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.

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 1).

Diagnostic accuracies (sensitivity and specificity)

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

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].

Table 1. Contingency table evaluating a test against a gold standard.

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

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.

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.

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.

```r
# 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 2). There is no requirement for consistency, which may benefit researchers performing

Table 2. 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

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.

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

1) The data must be input as a data frame (or matrix) with three columns.
2) The first column should contain values for test 1, the second for test 2 and the third column should contain the gold standard results.
3) The data need to be paired, i.e. the results on each row are for the same test sample or individual.
4) The observations on each individual row are independent, i.e. no biological or technical replicates, which violate the assumption of independence.

Failure to comply with rule 1 will result in an error. However, failure to comply with rules 2, 3 and 4 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 have been organised appropriately before implementing the 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.

Mathematical 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 3).

Sensitivity (Yu *et al.* interval):

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

where $z_{1-\alpha/2}$ is the 100(1 – α/2)th percentile of a standard normal distribution.

Specificity (Yu *et al.* interval):

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

where $z_{1-\alpha/2}$ is the 100(1 – α/2)th percentile of a standard normal distribution.

Table 3. 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

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

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

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

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

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

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

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

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

Positive likelihood ratio (approximation to the score method):

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

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:

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

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:

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

Replacing $x_1.$ with $x._1$ 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):

$$\frac{\tilde{n}\tilde{s}_0.\tilde{r}_0. + \frac{z^2_{1-\alpha/2}}{2}(\tilde{s}\tilde{s}_0. + \tilde{r}\tilde{r}_0. - 2\tilde{s}_0.\tilde{r}_0.) \pm z_{1-\alpha/2}\sqrt{\tilde{n}^2\tilde{s}_0.\tilde{r}_0.\left[\tilde{s}_0. + \tilde{r}_0. - \tilde{n}(1-\tilde{S}e_1)\tilde{S}p_1\right] + \frac{z^2_{1-\alpha/2}}{4}(\tilde{s}\tilde{s}_0. - \tilde{r}\tilde{r}_0.)^2}}{\tilde{r}_0.\left[\tilde{n}\tilde{s}\tilde{S}p_1 - z^2_{1-\alpha/2}(\tilde{s} - \tilde{r}_0.)\right]}$$

where $\tilde{s}_0. = s_0. + 0.5$, $\tilde{r}_0. = r_0. + 0.5$, $\tilde{s} = s + 1$, $\tilde{r} = r + 1$, $\tilde{n} = n + 2$, $\tilde{S}e_1 = \tilde{s}_1./\tilde{s}$ and $\tilde{s}p_1 = \tilde{r}_0./\tilde{r}$.

Regarding NLR_1, if the lower limit of the confidence interval is less than $\tilde{s}_0./(\tilde{n} - \tilde{r}_0.)$ or greater than $N\hat{L}R_1$ then the lower limit is given by:

$$\frac{\tilde{s}_0.\tilde{S}p_1 + \frac{z^2_{1-\alpha/2}}{2} - z_{1-\alpha/2}\sqrt{\frac{z^2_{1-\alpha/2}}{4} + \tilde{s}_0.(\tilde{S}p_1 + \tilde{S}e_1 - 1)}}{\tilde{s}\tilde{S}p_1^2 + z^2_{1-\alpha/2}},$$

Equally, if the upper limit is greater than $(\tilde{n} - \tilde{s}_{0.})/\tilde{r}_0.$ or less than $N\hat{L}R_1$ then the upper limit is given by:

$$\frac{\tilde{r}_0.(1-\tilde{S}e_1) + \frac{z^2_{1-\alpha/2}}{2} + z_{1-\alpha/2}\sqrt{\frac{z^2_{1-\alpha/2}}{4} + \tilde{r}_0.(1-\tilde{S}e_1 - \tilde{S}p_1)}}{\tilde{r}\tilde{S}p_1^2}.$$

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:

- `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.
- `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 1.

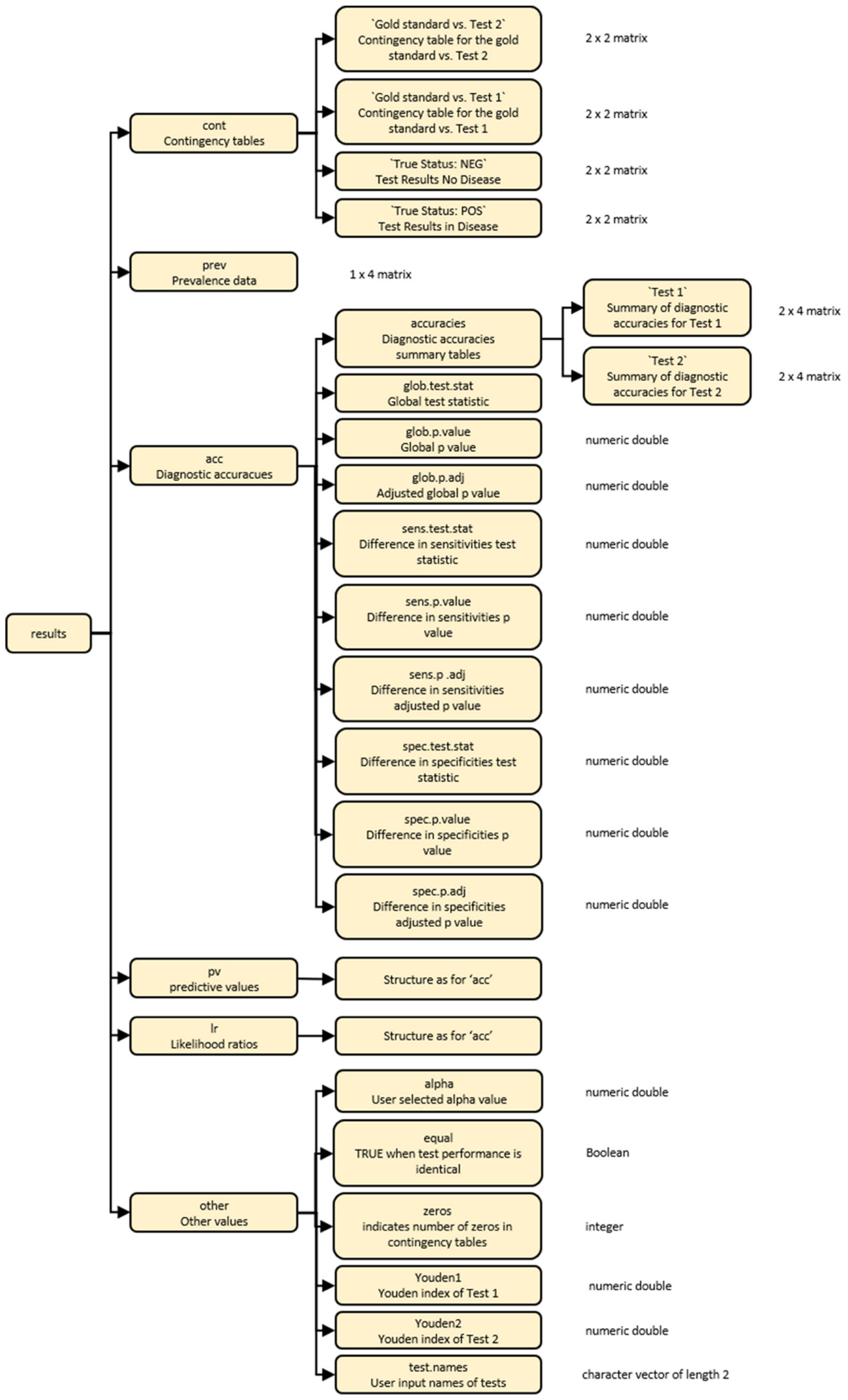

Figure 1. **A diagrammatic representation of the multilevel list output from the compareR() function.**

`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.

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.

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 4.

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 4).

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 5 and Figure 2A.

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 2B. 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].

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.

Table 4. 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	69.81	1.56	66.68	72.77						
compbdt	69.81	1.56	66.68	72.77						
Diagnostic accuracies: Test 1										
	Se	**SE**	**LCI**	**UCI**	**p**	**Sp**	**SE**	**LCI**	**UCI**	**p**
testCompareR	82.57	1.54	79.36	85.39	0	74.14	2.7	68.56	79.09	0.92
DTComPair	82.57	1.54	79.55	85.58	0	74.14	2.7	68.85	79.44	0.83
compbdt	82.57	1.54	79.36	85.39	0	74.14	2.7	68.58	79.09	0.99
Diagnostic accuracies: Test 2										
	Se	**SE**	**LCI**	**UCI**	**p**	**Sp**	**SE**	**LCI**	**UCI**	**p**
testCompareR	91.12	1.15	88.61	93.15		74.9	2.67	69.36	79.79	
DTComPair	91.12	1.15	88.86	93.38		74.9	2.67	69.67	80.14	
compbdt	91.12	1.15	88.61	93.15		74.9	2.67	69.36	79.79	
Predictive values: Test 1										
	PPV	**SE**	**LCI**	**UCI**	**p**	**NPV**	**SE**	**LCI**	**UCI**	**p**
testCompareR	88.07	1.36	85.17	90.50	0.37	64.78	2.75	59.25	69.98	0
DTComPair	88.07	1.36	85.41	90.73	0.37	64.78	2.75	59.39	70.18	0
compbdt	88.07	1.36	85.17	90.50	0.37	64.78	2.75	59.25	69.98	0
Predictive values: Test 2										
	PPV	**SE**	**LCI**	**UCI**	**p**	**NPV**	**SE**	**LCI**	**UCI**	**p**
testCompareR	88.07	1.36	85.17	90.50		64.78	2.75	59.25	69.98	
DTComPair	88.07	1.36	85.41	90.73		64.78	2.75	59.39	70.18	
compbdt	88.07	1.36	85.17	90.50		64.78	2.75	59.25	69.98	
Likelihood ratios: Test 1										
	PLR		**LCI**	**UCI**	**p**	**NLR**		**LCI**	**UCI**	**p**
testCompareR	3.19		2.61	3.95	0.37	0.23		0.2	0.28	0
DTComPair	3.19		2.59	3.93	0.37	0.23		0.2	0.28	0
compbdt	3.19		2.61	3.95	0.37	0.23		0.2	0.28	0
Likelihood ratios: Test 2										
	PLR		**LCI**	**UCI**	**p**	**NLR**		**LCI**	**UCI**	**p**
testCompareR	3.63		2.96	4.50		0.12		0.09	0.15	
DTComPair	3.63		2.94	4.48		0.12		0.09	0.15	
compbdt	3.63		2.96	4.50		0.12		0.09	0.15	

Table 5. 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

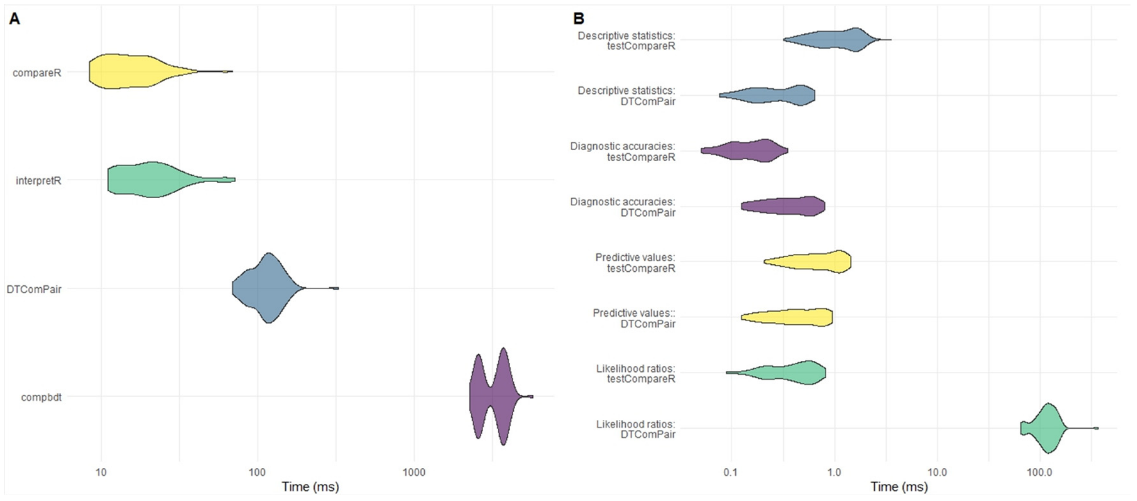

Figure 2. A) Comparison of computational time for testCompareR, DTComPair and compbdt. Each package was run 100 times. **B**) Comparison of testCompareR and DTComPair, by function. testCompareR compares the likelihood ratios more quickly than DTComPair because it uses a method based on an approximation 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.

```
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.

Discussion

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[5]. 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. 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 dramatically 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