testCompareR: a new R package to compare two dichotomous tests based upon paired data

Kyle J. Wilson, Marc Y.R. Henrion

## Abstract

In medicine the results of many tests are ultimately interpreted as positive or negative, given that we commonly want to determine if a patient’s status does or does not meet some condition, for example, whether they do or do not have a disease. These tests are generally described with common test metrics sensitivity and specificity and their derivative metrics positive and negative predictive value and likelihood ratio. The optimal methods for comparing two tests based upon these metrics have been the subject of much academic inquiry, but to date there is only one package for the open source statistical computing language R which addresses this issue. We have implemented up-to-date methods for the comparison of two diagnostic tests with dichotomous outcomes in a new R package. testCompareR rapidly performs a range of statistical tests on paired data in a single function call. Additionally, the package can provide a plain English readout, facilitating rapid interpretation of results. Here we present the package, discuss its strengths and limitations and provide a motivating example.

## Keywords

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

## Introduction

The determination of disease status based upon some test is a fundamental principle in medicine. Tests may be simple, 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 that’s presence or absence confirms or refutes the diagnosis, or by providing some cut off value above which the patient can be considered positive for the disease and below which they can be considered negative.

Dichotomous 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 test metrics sensitivity and specificity and their derivatives the predictive values and likelihood ratios. A brief explanation of each of these test metrics is provided in ‘Statistical methods’.

Unfortunately, evaluating whether one test performs better than another test is not as simple as just picking the test with the best metrics. As with all hypothesis testing we need to consider the possibility that the results obtained occurred due to random chance. Although the comparison of test metrics has been the subject of much academic enquiry, only one R package exists which performs this function.

We sought to develop a new R package which performs both descriptive and interential statistics on the whole range of test metrics using optimised statistical methods. The target users of this package are clinicians involved in the development and evaluation of diagnostic tests, not statisticians or computational scientists, and we therefore defined a list of features to maximise usability. Specifically, the new package should:

* 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 the user to select whether this matrix has margins displaying row and column sums
* provide the prevalence of the condition in question and a confidence interval based on the cohort studied
* allow the user to select which pairs of test metrics they are interested in (eg. 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 difference between the selected test metrics between 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 compares the results of two diagnostic tests with dichotomous outcomes based upon paired data.

## 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 1). 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.

Secondly, the structure of the data as presented to the package must conform to four rules: 1) the data structure must be either a data frame or matrix 2) the data structure must have three columns 3) the first column should contain values for test 1, the second for test 2 and the third column should contain the gold standard results 4) data should be paired, ie. every test should have been performed on each participant

Failure to comply with rules 1 and 2 will result in an error. However, failure to comply to result with rules 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 has been organised appropriately before implementing the analysis.

testCompareR does not automatically handle missing data. How to handle missing data is ultimately a decision for the research team. Removing cases with missing data or imputing missing data may introduce bias, especially when data is not missing at random. If in doubt, users of the package should discuss their individual situation with an experienced statistician.

## Statistical methods

The purpose of this software paper is to make clinicians aware of a new statistical tool which is now available to them and facilitate its use. Here we describe each of the test metrics before briefly summarising the mathematical basis of the tests used in the package. The statistical approach to each problem has been summarised in Table 2.

testCompareR can be considered an extension of the open-source compbdt program published by Roldán-Nofuentes, which is accompanied by a detailed review of the most up-to-date methods for constructing confidence intervals and performing hypothesis tests based on dichotomous test results in paired data. Our package offers some advantages over the original program which will be discussed in detail later, but the underlying statistical methodology is largely unchanged.

### Diagnostic accuracies

The fundamental descriptive statistics for tests with binary outcomes are the measures of diagnostic accuracy: sensitivity and specificity.

#### Sensitivity

Diagnostic sensitivity is the ability of a test to correctly identify disease in individuals whose disease status is positive. A test with high sensitivity has a high true positive rate with few false negatives.

Sensitivity can be described mathematically as:

#### Specificity

Diagnostic specificity is the ability of a test to correctly identify the absence of disease in individuals whose disease status is negative. A test with high sensitivity has a high true negative rate with few false positives.

Specificity can be described mathematically as:

### Predictive values

While diagnostic accuracies provide information about a tests ability to discriminate the true status of a patient, the predictive values describe the probability that an individuals true status matches the result of the test. Predictive values are influenced by the disease prevalence. As disease prevalence increases the positive predictive values increases and negative predictive value decreases. Conversely, as disease prevalence decreases the negative predictive value decreases and positive predictive value increases.

#### Positive predictive value

The positive predictive value is the probability that a patient with a positive test result truly has the disease.

Positive predictive value can be described mathematically as:

#### Negative predictive value

The negative predictive value is the probability that a patient with a negative test result is truly free from the disease.

Negative predictive value can be described mathematically as:

### Likelihood ratios

Likelihood ratios describe the likelihood of a test result in a patient with disease compared to the likelihood of the same result in a patient without disease. Likelihood ratios can be useful as they are less sensitive to differing disease prevalence.

#### Positive likelihood ratio

The positive likelihood ratio is the likelihood of a positive test result in a patient with disease compared to the likelihood of a positive result in a patient without disease, or the true positive rate divided by the false positive rate.

Positive likelihood ratio can be described mathematically as:

#### Negative likelihood ratio

The negative likelihood ratio is the likelihood of a negative test result in a patient with disease compared to the likelihood of a negative result in a patient without disease, or the false negative rate divided by the true negative rate.

Negative likelihood ratio can be described mathematically as:

### Constructing confidence intervals

#### Prevalence, diagnostic accuracies and predictive values

The prevalence, diagnostic accuracies and predictive values are all binomial proportions. Many options exist for constructing confidence intervals for binomial proportions. Yu et al. demonstrated that their modification of the Wilson interval has better asymptomic performance than other methods. The ‘Yu interval’ is implemented to construct confidence intervals for binomial proportions within testCompareR.

The Yu confidence interval for a given binomial proportion can be calculated as:

#### Likelihood ratios

Unlike the other test metrics likelihood ratios are not binomial proportions, but rather ratios of two, independent binomial proportions. Martín-Andrés & Álvarez-Hernández conducted comprehensive testing of multiple methods for constructing confidence intervals for ratios of independent binomial proportions. The best method was based on an approximation to the score method (after adding 0.5 to all the data). This method has been implemented in the testCompareR package.

Confidence intervals for the likelihood ratios can be calculated as:

### Hypothesis testing

#### Diagnostic accuracies

Using statistical simulation it has been demonstrated that the best methods for comparing diagnostic accuracies obtained from paired data vary depending on prevalence and total number of participants.

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:

. Where both conditions remain unmet the optimal method involves first testing the global null hypothesis:

The Wald teststatistic forms the basis of this test. If the global null hypothesis is maintained then neither difference can be considered significant. When the global null is rejected then individual hypothesis tests are performed to determine the cause of significance. When total number of participants is less than or equal to 100, or greater than or equal to 1000, then the Wald statistic applies. In cases where total number of participants is between 100 and 1000, exclusive, then McNemar’s test is used. 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 the global hypothesis test:

If the global hypothesis is rejected, the causes of significance are investigated using the weighted generalised score statistic, as described by Kosinski.

#### Likelihood ratios

Unsurprisingly, 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 statistic. Where the global null is rejected the cause of significance is determined by individual hypothesis tests as previously described.

## Package implementation

The package is made up 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 structured as per ‘Data preparation’. A whole gamut of internal functions then ensure data is correctly coded, before calculating output values according to the methodologies described in ‘Statistical methods’. A range of option parameters allow users to customise the output:

* alpha An alpha value. Defaults to 0.05.
* margins A Boolean value indicating whether the contingency tables should have margins containing summed totals of rows and columns.
* multi\_corr Method for multiple comparisons. Uses p.adjust.methods.
* cc A Boolean value indicating whether McNemar’s test should be applied with continuity correction.
* dp Number of decimal places of output in summary tables. Defaults to
* sesp A Boolean value indicating whether output should include sensitivity and specificity.
* ppvnpv A Boolean value indicating whether output should include positive and negative predictive values.
* plrnlr A Boolean value indicating whether output should include positive and negative likelihood ratios.
* test.names A vector of length two giving the names of the two different binary diagnostic tests. This argument is not relevant when testing a single binary diagnostic test.
* ... Rarely needs to be used. Allows additional arguments to be passed to internal functions.

The output from the compareR function is a multilevel list object of class compareR. For those wishing to 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.](development/list_structure.png)

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 data sets. Like compareR(), summariseR() allows flexible input, which can prevent researchers having to manually re-code their data.

Additionally, the dataframeR function takes numerical arguments representing the eight fundamental values which are required to run compareR and creates a data frame which can be supplied to compareR. This function facilitates secondary data analysis and meta-analysis.

Many internal functions facilitate the actions of the exported functions available to the user. A detailed description of the internal functions is beyond the scope of this paper, but each has a documentation file accessible within the package.

## Examples

To demonstrate the use of the testCompareR package we will utilise the Coronary Artery Surgery Study (cass) data set which is included with the package. This data set 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). It has become a standard for testing in statistical research regarding test metrics.

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, and angio, which reports the outcome of the gold standard test. Here, we can see that the data is already coded as zeros and ones.

head(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

tail(cass)

exercise cp angio  
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 described in ‘Package implementation’. To avoid an unnecessary lengthy output in the example we have used the parameters ppvnpv and plrnlr. Setting them to FALSE 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

Our package is elegant in its simplicity. Several parameters permit customisation of the output, but they are not elaborated here as they are beyond the scope of this introductory article, and are not essential to understand the workings of the package. Further details can be found within the package vignette, which contains examples of all modifiableq  
parameters.

## Validation and performance evaluation

### Features

We compared the features of the testCompareR package to the DTComPair package and to the compbdt() function, which are both publicly available.

Evaluation of the features of each package is based on a standard hypothetical workflow whereby a researcher has their data stored in a database or spreadsheet which can then be imported into R using read.csv(). The resulting data frame would then be supplied to each package.

When evaluating the features of each package we considered the following:

1. Are pre-processing steps required before the user can supply their data to the package / function?
2. Can the package / function accept flexible input data formatting?
3. How many function calls are required to return diagnostic accuracies, predictive values and likelihood ratios for two tests, as well as compare differences in each of the three pairs of test metrics?
4. If required, can individual test results be accessed via indexing for post-processing?
5. Can the package / function handle NAs?
6. Does the package / function allow adjustments for multiple testing?
7. Does the package / function provide an interpretation of the results?
8. Is the package available on CRAN?

The results of our comparison of features is displayed in Table 3.

By reducing the number of steps required to produce results, allowing flexible input and having options to access results by indexing or as a plain English readout the testCompareR package facilitates rapid statistical analysis.

### Statistical performance

As the compbdt function provides the statistical methods by which the testCompareR package arrives at its results, there is very little difference between them in terms of statistical performance. However, there are several important differences between testCompareR and DTComPair.

Firstly, Yu et al. describe a modified version of the score interval for computing confidence intervals from binomial proportions which has been implemented in testCompareR. Simulation studies demonstrated superior performance of the Yu interval when compared to other common intervals, including the interval proposed by Agresti & Coull which is the basis of the confidence intervals in the DTComPair package. Specifically, the Agresti & Coull interval is much too conservative at p values close to zero.

Secondly, concerning confidence intervals for the likelihood ratios, testCompareR uses the best method for calculating confidence intervals based on the ratio of independent binomial proportions as determined by a comprehensive simulation study which evaluated 73 methods for constructing confidence intervals. The Simel interval used in the DTComPair package, has suboptimal coverage when sample sizes are not large.

Finally, the DTComPair package uses individual hypothesis tests to compare different test metrics. By contrast, the testCompareR package uses global hypothesis tests to test each pair of test metrics, before evaluating the cause of significance if a difference is identified. Simulation studies have demonstrated the superiority of global hypothesis testing when compared to individual hypothesis testing of test metrics.

#### Comparing results for diagnostic accuracies

We compared the output of each of the functions to validate the results, using the diagnostic accuracies as a test case. All three functions have highly similar performance when comparing the diagnostic accuracies of the cass data set.

|  | Se | LCI | UCI | p | Sp | LCI | UCI | p |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| testCompareR | 82.57 | 79.36 | 85.39 | 0 | 74.14 | 68.56 | 79.09 | 0.99 |
| DTComPair | 82.57 | 79.55 | 85.58 | 0 | 74.14 | 68.85 | 79.44 | 0.83 |
| compbdt | 82.57 | 79.36 | 85.39 | 0 | 74.14 | 68.58 | 79.09 | 0.99 |

|  | Se | LCI | UCI | p | Sp | LCI | UCI | p |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| testCompareR | 91.12 | 88.61 | 93.15 | NA | 74.9 | 69.36 | 79.79 | NA |
| DTComPair | 91.12 | 88.86 | 93.38 | NA | 74.9 | 69.67 | 80.14 | NA |
| compbdt | 91.12 | 88.61 | 93.15 | NA | 74.9 | 69.36 | 79.79 | NA |

|  | PPV | LCI | UCI | p | NPV | LCI | UCI | p |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| testCompareR | 88.07 | 85.17 | 90.50 | 0.37 | 64.78 | 59.25 | 69.98 | 0 |
| DTComPair | 88.07 | 85.41 | 90.73 | 0.37 | 64.78 | 59.39 | 70.18 | 0 |
| compbdt | 88.07 | 85.17 | 90.50 | 0.37 | 64.78 | 59.25 | 69.98 | 0 |

|  | PPV | LCI | UCI | p | NPV | LCI | UCI | p |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| testCompareR | 89.36 | 86.70 | 91.56 | NA | 78.49 | 73.02 | 83.15 | NA |
| DTComPair | 89.36 | 86.93 | 91.78 | NA | 78.49 | 73.40 | 83.57 | NA |
| compbdt | 89.36 | 86.70 | 91.56 | NA | 78.47 | 73.02 | 83.15 | NA |

|  | 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 | 319.34 | 259.39 | 393.13 | 0.37 | 23.51 | 19.5 | 28.35 | 0 |
| compbdt | 3.19 | 2.61 | 3.95 | 0.37 | 0.23 | 0.2 | 0.28 | 0 |

|  | PLR | LCI | UCI | p | NLR | LCI | UCI | p |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| testCompareR | 3.63 | 2.96 | 4.50 | NA | 0.12 | 0.09 | 0.15 | NA |
| DTComPair | 363.09 | 294.24 | 448.06 | NA | 11.86 | 9.11 | 15.44 | NA |
| compbdt | 3.63 | 2.96 | 4.50 | NA | 0.12 | 0.09 | 0.15 | NA |

### Computational performance

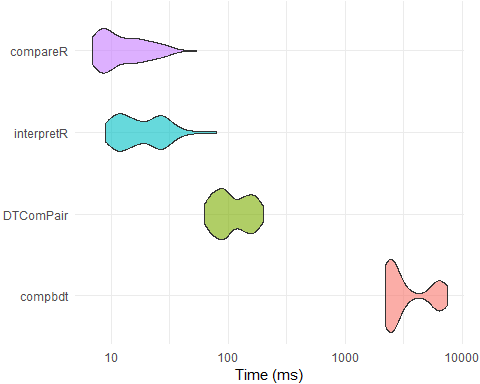
To evaluate the performance of the package we used testCompareR to compute the results using the Coronary Artery Surgery Study data set which is included in the package (cass). This data set looks at exercise stress testing and history of chest pain as two tests for coronary artery disease as determined by coronary angiography (the gold standard). It has become a standard for testing in statistical research regarding test metrics.

We used the same data to compute the same results using the DTComPair and compbdt packages, after a small amount of pre-processing to make the data conform to the requirements of each package. Pre-processing steps were considered in an attempt to replicate the steps a researcher might take to produce their results.

Using the microbenchmark package this procedure was repeated 100 times and the time elapsed during each function call was recorded.

We found that compareR() from the testCompareR package computed the results more quickly than DTComPair and the combination of compareR and interpretR was faster than both alternatives.

expr time   
 compareR :100 Min. :6.968e+06   
 interpretR:100 1st Qu.:1.394e+07   
 DTComPair :100 Median :6.404e+07   
 compbdt :100 Mean :9.318e+08   
 3rd Qu.:7.029e+08   
 Max. :7.397e+09



Further testing demonstrated that the cause of the difference in the DTComPair package is the method for comparing the likelihood ratios. The method used by DTComPair is based on logistical regression, whereas the method used by testCompareR uses a method based on an approximation of the score statistic, which is simpler to compute requiring only solving of a second degree equation.

expr time   
 describe:100 Min. : 79901   
 h.acc :100 1st Qu.: 111102   
 h.pv :100 Median : 161250   
 h.lr :100 Mean : 22226549   
 3rd Qu.: 15857302   
 Max. :236434200

### Data validation

Data validation is provided by a range of custom error messages which have been comprehensively tested use the testthat package. The suite of 557 tests runs in 1.4 seconds using R 4.3.0 on Windows 10 Enterprise OS with 16GB RAM and 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz.

## Discussion

Despite the common use of binary diagnostic tests only one package, DTComPair, provides methods to compare the test metrics between two tests with dichotomous outcomes using paired data. This package requires the user to be reasonably computationally literate as several function calls are necessary to extract the outputs that would normally be published following a trial to compare the performance of two tests. Additionally, though the package implements well-loved 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. Additionally, here we have shown that the newer methods for comparing the likelihood ratios between two tests are more computationally efficient.

By re-structuring the internal mechanisms of the open-source compbdt program we have dramatically increased computational speed while providing additional features - the testCompareR user can choose whether to receive their output in list form, allowing them to access individual elements via indexing, or as a plain English summary, facilitating rapid interpreation of the results.

Recently, the COVID-19 pandemic brought to our attention the importance of pandemic preparedness. 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.

## Data & software availability

The testCompareR package is available through the Comprehensive R Archiving Network (CRAN) and the source code is available on Github. The software is distributed via the GPL-2 license.

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).

## Reporting guidelines

We identified no specific reporting guidelines for software papers in health research.

## Consent

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

## Author contributions

Kyle J. Wilson: conceptualisation, methodology, software, validation, writing - original draft preparation.

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

## Competing interests

The authors declare no competing interests.

## Grant information

This work is supported by Wellcome Trust grants 222530/Z/21/Z and 206545/Z/17/Z.

## 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