dtComb: A Comprehensive R Package for Combining Two Diagnostic Tests

Serra İlayda YERLİTAŞ [1,2], Serra Berşan GENGEÇ [2], Gözde ERTÜRK ZARARSIZ[1,2], Selçuk KORKMAZ[3] and Gökmen ZARARSIZ[1,2†]

[1] *Erciyes University Faculty of Medicine Department of Biostatistics, Kayseri, TURKEY*

[2] *Erciyes University Drug Application and Research Center (ERFARMA) , Kayseri, TURKEY*

[3] *Trakya University Faculty of Medicine Department of Biostatistics, Edirne, TURKEY*

June 10, 2022

Abstract

dtComb is a comprehensive R package developed to combine two different diagnostic tests and offers users an extensive range of distinct methods brought together. Using the dtComb package, researchers are able to apply standardization to their data and combine these diagnostic tests with 143 combination methods available in the package by loading the diagnostic tests and the dataset containing the reference test. Combination methods included in the package are gathered under 4 main groups: Linear Combination Methods (`linComb`), Nonlinear Combination Methods (`nonlinComb`), Mathematical Operators (`mathComb`) and Machine Learning Algorithms (`mlComb`). Within the scope of Linear Combination approaches there are 8 combination methods gathered and applied from literature. Nonlinear Combination Methods include several statistical approaches such as polynomial regression, penalized regression methods and splines. In addition to these methods, the interactions between the diagnostic tests can also be incorporated to the combination model. Arithmetic operators are included in the group of Mathematical Operators as well as 8 `distance measures` that can be used according to the data structure. The final group is Machine Learning Algorithms, which consist of 113 machine learning algorithms from the caret package that are suitable for the dtComb data structure.

For data standardization, five different standardization methods have been prepared for linear, non-linear combination methods and mathematical operators in the package. These methods allows users to standardize the data according to `range`, `z score`, `t score`, `mean`, and `standard deviation` calculated from the loaded dataset. For the machine learning approaches, the `preprocessing` methods included in the caret package can be used within the dtComb package for standardization.

In terms of `resampling` methods, Cross-Validation, Bootstrapping, and Repeated Cross-Validation are available to be used with linear and nonlinear combination methods for model building and performance evaluation. For ML algorithms, the `resampling` methods in the caret package are applicable.

Finally, using the models built with these aforementioned functions, the `predComb` function, predicts class labels and combination scores of a given set of test samples.

The dtComb package is user-friendly, effortless, and is currently the most comprehensive single package developed in the literature to combine diagnostic tests. This vignette has been created to guide researchers on how to use this package.

dtComb version: 0.99.6

1 Introduction

Diagnostic tests are widely used for purposes of distinguishing diseases from each other and making the correct diagnosis for the patients. Therefore, the role of the diagnostic tests in making decisions

is crucial. In addition to having an essential role in medical diagnosis, these tests also contribute to planning the proper treatment while reducing treatment costs. The diagnostic accuracy performance and reliability of these diagnostic tests are taken into account when making these tests widely available. For some conditions, more than one diagnostic test may be available. Some of these diagnostic tests may even replace existing methods because they have better performance. In many studies, new diagnostic test performance measures with superior performance were obtained by using multiple diagnostic test instead of one. Various approaches to combining diagnostic tests are available in the literature. The *dtComb* package includes diverse combination methods, data standardization, and resampling methods. This vignette will demonstrate how to combine two different diagnostic tests with different combination methods. *dtComb*package is loaded as below:

```
library(dtComb)
```

2 Preparing the input data

The methods available in this package are structured to require a `DataFrame` with two categories (negative/positive, present/absent) as the results of a reference test used in the precision of the disease and a continuous quantitative structure for the diagnostic tests. In this demonstration, we will be using the dataset named exampleData1 that is included in this package. The exampleData1 dataset contains data from patients admitted to the Erciyes University Medical Faculty's General Surgery Department with complaints of abdominal pain. Dataset includes the leukocyte counts and D-dimer levels of 225 patients (115 females, 110 males) of two groups that indicates the results of the reference test. The first group had 115 $(51,1\%)$ patients who needed immediate laparotomy, and the second group had 110 $(49,9\%)$ patients who did not need immediate laparotomy. Conventional treatment is assessed according to this grouping, and the patients operated on based on their postoperative pathologies are assigned to the first group. In contrast, the patients with a negative laparotomy are assigned to the second group [1].
The dataset exampleData1 is called using the `data` function to set up the model. The data set is structured as follows.

```
data(exampleData1)
head(exampleData1)

##    group ddimer log_leukocyte
## 1 needed   8.09          5.52
## 2 needed   5.16          4.43
## 3 needed   8.90          5.20
## 4 needed  10.17          5.39
## 5 needed   1.93          5.09
## 6 needed   3.63          4.68
```

In this demonstration a training set and a test set will be used to show the capabilites of this package. The training set will consist of 75% of the data set and will be used to train classification models as well as to compare different model performances. The remaining of the data set will be saved as a test set to later demonstrate the prediction functionality using the trained model parameters of selected classification models. The `train` set is constructed as below, and the `test` set will be set up later on.

```
# # train set from the exampleData1
set.seed(2128)
inTrain <- caret::createDataPartition(exampleData1$group, p = 3/4, list = FALSE)
trainData <- exampleData1[inTrain, ]
head(trainData)

##    group ddimer log_leukocyte
## 2 needed   5.16          4.43
```

```
## 3 needed   8.90          5.20
## 4 needed  10.17          5.39
## 5 needed   1.93          5.09
## 6 needed   3.63          4.68
## 7 needed   3.12          5.20
```

When the contents of the `train` set is examined it can be seen that out of 170 observations in this set, 83 patients require laparotomy, while 87 patients do not require laparotomy according to the group column.
Functions used for training are designed to take diagnostic tests and reference test results as two different inputs. In this step, the training set obtained from the sample data set is divided into diagnostic test results (`markers`) and reference test results (`status`). During this division process, the referencce test results (`status`) are transformed into factor variables.

```
markers <- trainData[, -1]
status <- factor(trainData$group, levels = c("not_needed", "needed"))
```

3 Avaible Methods

There are various methods for combining diagnostic tests. One of the extensively used method is comparing the two diagnostic tests. However, models with notably higher accuracy can be obtained using statistical methods and machine learning algorithms. The *dtComb* package includes 143 methods for combining diagnostic tests. These methods are divided into linear methods, non-linear methods, mathematical operators, and machine learning algorithms.

3.1 Linear Combination Methods:

- **Scoring based on Logistic Regression (`scoring`)**: Combination score is obtained using the slope values of the relevant logistic regression model, slope values are rounded to the number of digits taken from the user. [2].

$$l = \log_b\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$
$$Combination\ Score = \beta_1 x_1 + \beta_2 x_2$$

- **Su & Liu's method (`SL`)**: Su and Liu combination score is obtained by using Fisher's discriminant function under the assumption of multivariate normal distribution model and proportionate covariance matrices [3].

$$Contol\ group = X \sim N(\mu_x, \sum)$$
$$Disease\ group = Y \sim N(\mu_y, \sigma^2 \sum)$$
$$Fisher's\ discriminant\ coefficient = (\alpha, \beta) \propto (\mu_y - \mu_x)^T \sum^{-1}$$
$$Combination\ Score = \alpha x_1 + \beta x_2$$

- **Logistic Regression (`logistic`)**: Combination score obtained by fitting a logistic regression model [4].

$$l = \log_b\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

- **Min-Max method (`minmax`)**: Linearly combines the minimum and maximum values of the markers by finding a parameter λ that maximizes corresponding Mann-Whitney statistic. [5].

$$Control\ group : X_i = (X_1 i, X_2 i), i = 1, 2, ..., n$$
$$Disease\ group : Y_j = (Y_1 j, Y_2 j), j = 1, 2, ..., m$$
$$maximize\ W(\lambda) = \left(\frac{1}{mn}\right) \sum_{i=1}^{n} \sum_{j=1}^{m} I(Y_{j,max} + \lambda Y_{j,min} > X_{i,max} + \lambda X_{i,min})$$
$$Combination\ Score = x_{max} + \lambda x_{min}$$

- **Pepe & Thompson's method (`PT`)**:Pepe and Thompson combination score obtained by proportioning the slope values the relevant logistic regression model to obtain the parameter λ [6].

$$l = \log_b \left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$
$$\lambda = \beta_2 / \beta_1$$
$$Combination\ Score = x_1 + \lambda x_2$$

- **Pepe, Cai & Langton's method (`PCL`)**: Pepe, Cai and Langton combination score obtained by using AUC as the parameter of a logistic regression model [7].

$$Control\ group : X_i = (X_1 i, X_2 i), i = 1, 2, ..., n$$
$$Disease\ group = Y_j = (Y_1 j, Y_2 j), j = 1, 2, ..., m$$
$$maximize\ W(\lambda) = \left(\frac{1}{mn}\right) \sum_{i=1}^{n} \sum_{j=1}^{m} I(Y_{1j} + \lambda Y_{2j} > X_{1i} + \lambda X_{2i}) + \left(\frac{1}{2}\right) I(Y_{1j} + \lambda Y_{2j} = X_{1i} + \lambda X_{2i})$$
$$Combination\ Score = x_1 + \lambda x_2$$

- **Minimax approach (`minimax`):** Combination score obtained with the Minimax procedure; t parameter is chosen as the value that gives the maximum AUC from the combination score. [8].

$$Contol\ group : X_i = (X_{1i}, X_{2i}), i = 1, 2, ..., n$$
$$Disease\ group = Y_j = (Y_{1j}, Y_{2j}), j = 1, 2, ..., m$$
$$(b_1, b_2) = [t \sum_{D} + (1-t) \sum_{C}]^{-1} (\mu_D - \mu_C)$$
$$Combination\ Score = b_1 x_1 + b_2 x_2$$

- **Todor & Saplacan's method (`TS`)**: Combination score obtained by using the trigonometric functions of the Θ value that optimizes the corresponding AUC [9].

$$Combination\ Score = sin(\theta) x_1 + cos(\theta) x_2$$

3.2 Nonlinear Combination Methods:

- **Logistic Regression with Polynomial Feature Space (`polyreg`)**: The method builds a logistic regression model with the feature space created and returns the probability of a positive event for each observation.

- **Ridge Regression with Polynomial Feature Space (`ridgereg`)**: Ridge regression is a shrinkage method used to estimate the coefficients of highly correlated variables and in this case the polynomial feature space created from two biomarkers. For the implementation of the method, `glmnet` library is used with two functions: `cv.glmnet` to run a cross validation model to determine the tuning parameter λ and `glmnet` to fit the model with the selected tuning parameter [10].

- **Lasso Regression with Polynomial Feature Space (`lassoreg`)**: Lasso regression is also a shrinkage method with one difference is that at the end this method returns the coefficients of some features as 0, makes this method useful for feature elimination as well. The implementation is similar to Ridge regression, cross validation for parameter selection and model fit are implemented with `glmnet` library [10].
- **Elastic Net Regression with Polynomial Feature Space (`elasticreg`)**: Elastic Net regression is obtained by combining the penalties of Ridge and Lasso regression to get the best of both models. The model again includes a tuning parameter λ as well as a mixing parameter α taken form the user which takes a value between 0 (ridge) and 1 (lasso) to determine the weights of the loss functions of Ridge and Lasso regressions [10]
- **Splines (`splines`)**: With the applications of regression models in a polynomial feature space the second non-linear approach to combining biomarkers comes from applying several regression models to the dataset using a function derived from piecewise polynomials. Splines are implemented with degrees of freedom and degrees of the fitted polynomials taken from the user. For the implementation `splines` library is used to build piecewise logistic regression models with base splines [11].
- **Generalized Additive Models with Smoothing Splines and Generalized Additive Models with Natural Cubic Splines (`sgam ve nsgam`)**: In addition to the basic spline structure, Generalized Additive Models are applied with natural cubic splines and smoothing splines using the `gam` library [10] in R [12].

One of the non-linear approaches is to incorporate any interaction that may exist between the two diagnostic tests into the model built. Incorporating the interaction of variables to a regression model has benefits when there is a correlation between the diagnostic tests. To include interaction while building the model, the `include.interact` option will be selectes as `TRUE` in the nonlinComb function.

3.3 Mathematical Operators:

- **Arithmetic Operators** : `add`, `subtract`, `multiply` ve `divide` methods represent basic arithmetic operators.
- **Distance Measures**: The distance measures included in the package are `Euclidean`, `Manhattan`, `Chebyshev`, `Kulczynski d`, `Lorentzian`, `Avg`, `Taneja`, and `Kumar-Johnson`. The mathematical calculation methods for these measures are as follows. [13].
  - **Euclidean**(`euclidean`) $= \sqrt{\sum_{i=1}^{N} |P_i - Q_i|^2}$
  - **Manhattan**(`manhattan`) $= \sum_{i=1}^{N} |P_i - Q_i|$
  - **Chebyshev**(`chebyshev`) $= max|P_i - Q_i|$
  - **Kulczynski d**(`kulczynski-d`) $= \frac{\sum_{i=1}^{N} |P_i - Q_i|}{\sum_{i=1}^{N} min(P_i, Q_i)}$
  - **Lorentzian**(`lorentzian`) $= \sum_{i=1}^{N} \ln(1 + |P_i - Q_i|)$
  - **Avg**(`avg`) $= \frac{\sum_{i=1}^{N} |P_i - Q_i| + max|P_i - Q_i|}{2}$
  - **Taneja**(`taneja`) $= \sum_{i=1}^{N} \frac{P_i + Q_i}{2} . \log(\frac{P_i + Q_i}{2\sqrt{P_i . Q_i}})$
  - **Kumar-Johnson**(`kumar-johnson`) $= \sum_{i=1}^{N} \frac{(P_i^2 - Q_i^2)^2}{2.(P_i . Q_i)^{3/2}}$
- **Exponential functions**: These methods, in which one of the two diagnostic tests is taken as base and the other as an exponent, are indicated by the names `baseinexp`($markers_1^{markers_2}$) and `expinbase` ($markers_2^{markers_1}$).

Another option given is to apply a transformation on the markers before using the methods available in mathematical operators. The types of transformations that can applied are *logarithmic* (`log`), *exponential* (`exp`), *sine* (`sin`), and *cosine* (`cos`). The `power.transform` option can also be used to optimize the **AUC** on biomarkers. When the `power.transform` option is selected as `TRUE`, the power of the biomarkers are taken in increments of 0.1 step size within the [-3, 3] range and the method chosen by the user is applied to the diagnostic tests, the point with maximum **AUC** is returned and chosen for the final model.

3.4 Machine Learning Algorithms:

113 machine learning algorithms available in the `caret` library, which are used to train classification models using machine learning algorithms and make predictions using these models, are used within the scope of `mlComb` function [14].

The `availableMethods` function can be used to see the machine learning algorithms included in the package.

4 Standardization

Standardization is one of the important steps in data analysis when diagnostic tests with different units are used. It is better to have the data in the same unit or range in most cases. Standardization is provided as an option that can be applied with all models, although some combination methods available in the *dtComb* package require standardization by default. Standardization methods have been prepared for `linComb`, `nonlinComb`, and `mathComb` functions without depending on other libraries, and the details of the methods used for standardization are given below:

- **range**: Standardization to a range between 0 and 1
- **zScore**: Standardization using z scores with mean equals to 0 and standard deviation equals to 1
- **tScore**: Standardization using T scores. The range varies between usually 20 and 80
- **mean**: Standardization with sample mean equals to 1
- **deviance**: Standardization with sample standard deviation equals to 1

The standardization process for the `mlComb` function is performed using the `preprocessing` function in the `caret` package.

5 Resampling

The resampling options for parameter selection in linear and non-linear combinations are specified with the `resample` argument. Cross-Validation (`cv`), Repeated Cross-Validation (`cv`), and Bootstrapping (`boot`) methods are given as resampling options in these combination models. For the Cross-validation option, `nfolds` is given to determine the number of groups to be created; for the Repeated cross validation option `nrepeat` is used to specify the number of repetitions, and for the Bootstrapping option `niters` is given to determine the number of subgroups to be created.

Resampling for the `mlComb` function is performed using the `resampling` function included in the `caret` package. Detailed information is available in the relevant section of the `caret` package [14].

6 Cutoff and Direction

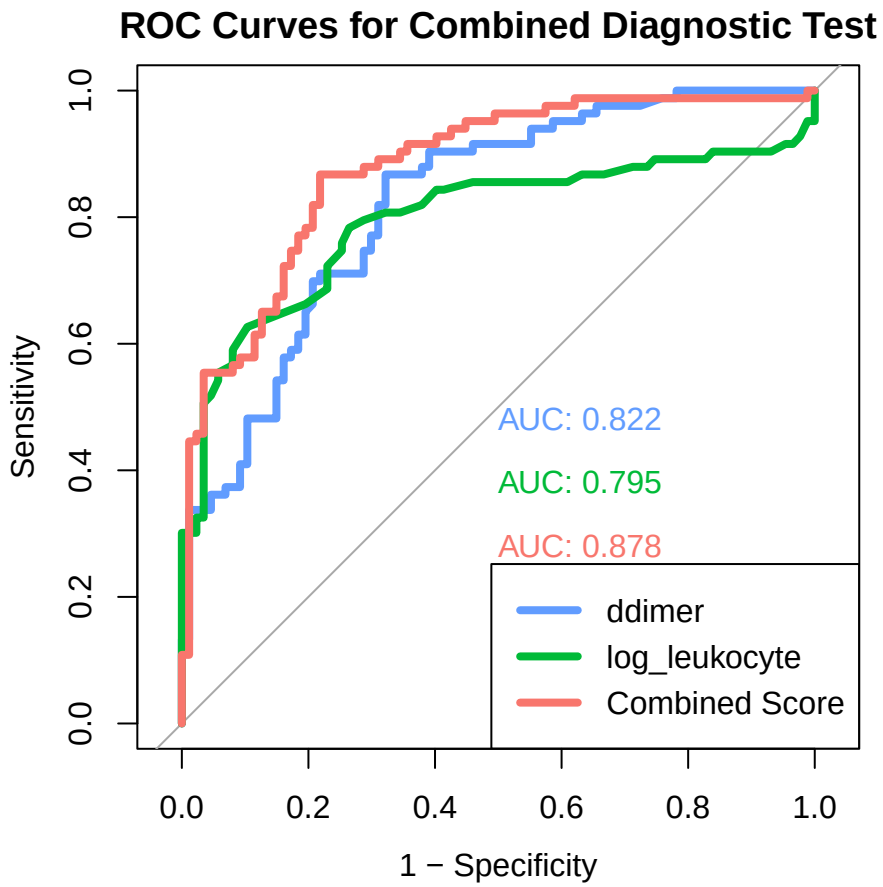
AUC values and ROC curves are considered when evaluating the model performance in the training process. The `direction` argument determines the direction of the ROC curve and is given as an input to the relevant function. Its default value is set to `auto`. The cutoff point in the ROC curve drawn to determine the AUC value is controlled by the `cutoff.method` argument. `youden` and `roc01` method options are available for this argument [15].

7 Model building

dtComb has 4 different functions (`linComb`, `nonlinComb`, `mlComb`, `mathComb`) to build and evaluate models. These functions establish the appropriate model in line with the methods chosen by the user and present the performance to the user. For the building of the model, previously prepared markers and status variables, the state where the `event` to be classifed occurred, the method to be used, and the arguments subject to the method are given as input to the functions. One function from each group is prepared as examples that will be built using the specifications below:

- For linear combination function, `scoring` (`ndgits = 2`) method will be used together with `resample = cv` (`nfold = 5`) and `range` as the standardization method
- For non-linear combination function, `lassoreg` (`include.interact = TRUE`) method will be used together with `boot` (`niters = 10`) as the resampling method
- For machine learning combination function, `knn` method will be used together with `repeatedcv` (`nfolds = 10, nrepeats = 5`) as the resampling method
- For the mathematical operators function, `distance` (`distance = \euclidean"`) method will be used.

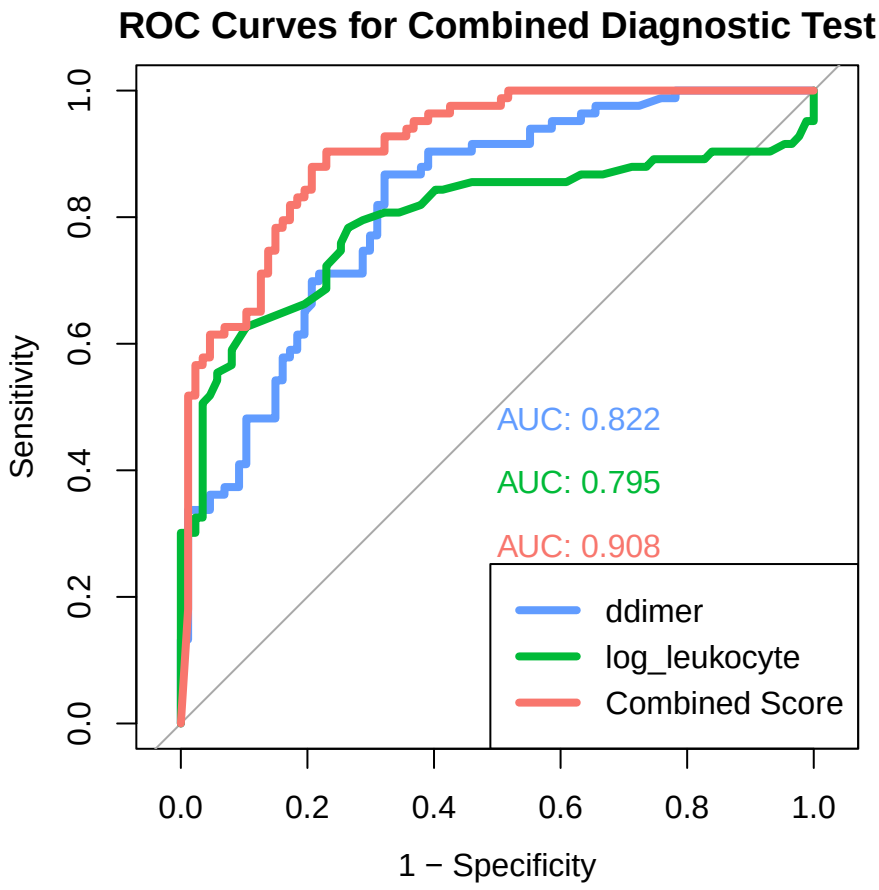
```r
set.seed(2128)

#linComb Function
fit.lin <- linComb(markers = markers,
                        status = status,
                        event = "needed",
                        method = "scoring",
                        resample = "cv",
                        standardize = "range",
                        ndigits = 2, direction = "auto",
                        cutoff.method = "youden")
```



```
## Method: scoring
## Samples: 170
## Markers: 2
## Events: not_needed, needed
## Standardization: range
## Resampling: cv (nfolds: 5)
##  Kappa       Accuracy
##  0.6476441   0.8235294
##
## Area Under the Curves of markers and combined score:
##                      AUC     SE.AUC LowerLimit UpperLimit         z      p.value
## ddimer         0.8221853 0.03136528  0.7607105  0.8836601 10.272036 9.419585e-25
## log_leukocyte 0.7950422 0.03718164  0.7221676  0.8679169  7.935159 2.102258e-15
## Combined       0.8781332 0.02617685  0.8268275  0.9294389 14.445332 2.682607e-47
##
## Area Under the Curve comparison of markers and combined score:
##   Marker1 (A)   Marker2 (B)   AUC (A)   AUC (B)      |A-B|  SE(|A-B|)         z
## 1    Combined        ddimer 0.8781332 0.8221853 0.05594793 0.02324285 2.4071030
## 2    Combined log_leukocyte 0.8781332 0.7950422 0.08309098 0.03097344 2.6826531
## 3      ddimer log_leukocyte 0.8221853 0.7950422 0.02714305 0.04705981 0.5767779
##      p-value
## 1 0.01607963
## 2 0.00730407
## 3 0.56408952
##
## Confusion matrix:
##           Outcome +    Outcome -      Total
## Test +           72           19         91
## Test -           11           68         79
## Total            83           87        170
##
## Point estimates and 95% CIs:
## -------------------------------------------------------------
## Apparent prevalence *                 0.54 (0.46, 0.61)
## True prevalence *                     0.49 (0.41, 0.57)
## Sensitivity *                         0.87 (0.78, 0.93)
## Specificity *                         0.78 (0.68, 0.86)
## Positive predictive value *           0.79 (0.69, 0.87)
## Negative predictive value *           0.86 (0.76, 0.93)
## Positive likelihood ratio             3.97 (2.65, 5.96)
## Negative likelihood ratio             0.17 (0.10, 0.30)
## False T+ proportion for true D- *     0.22 (0.14, 0.32)
## False T- proportion for true D+ *     0.13 (0.07, 0.22)
## False T+ proportion for T+ *          0.21 (0.13, 0.31)
## False T- proportion for T- *          0.14 (0.07, 0.24)
## Correctly classified proportion *     0.82 (0.76, 0.88)
## -------------------------------------------------------------
## * Exact CIs
```

```
#nonlinComb Function
fit.nonlin <- nonlinComb(markers = markers,
                         status = status,
                         event = "needed",
                         method = "lassoreg",
```

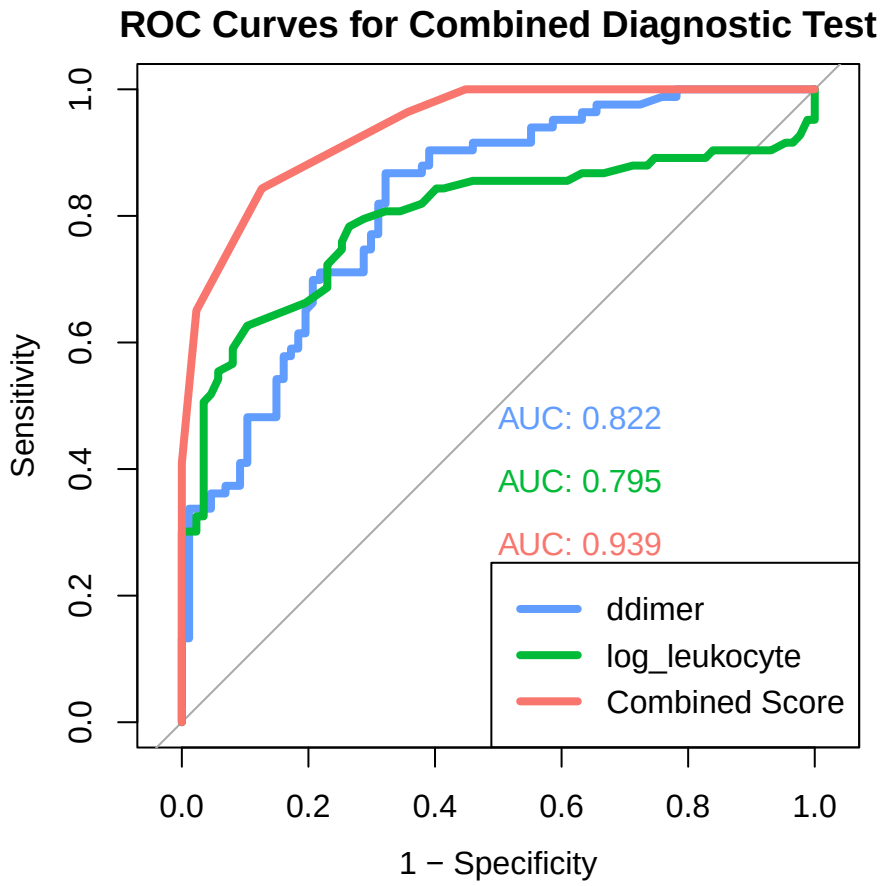
```
                        include.interact = "TRUE",
                        resample = "boot",
                        direction = "auto",
                        cutoff.method = "youden")
```




```
## Method: lassoreg
## Samples: 170
## Markers: 2
## Events: not_needed, needed
## Standardization: none
## Resampling: boot (niters: 10)
##  Kappa       Accuracy
##  0.6714976   0.8352941
##
## Area Under the Curves of markers and combined score:
##                     AUC     SE.AUC LowerLimit UpperLimit         z      p.value
## ddimer        0.8221853 0.03136528  0.7607105  0.8836601 10.272036 9.419585e-25
## log_leukocyte 0.7950422 0.03718164  0.7221676  0.8679169  7.935159 2.102258e-15
## Combined      0.9082537 0.02162706  0.8658654  0.9506420 18.876983 1.763788e-79
##
## Area Under the Curve comparison of markers and combined score:
##   Marker1 (A)   Marker2 (B)   AUC (A)   AUC (B)      |A-B|  SE(|A-B|)         z
## 1    Combined        ddimer 0.9082537 0.8221853 0.08606841 0.01850002 4.6523421
## 2    Combined log_leukocyte 0.9082537 0.7950422 0.11321147 0.03858083 2.9343972
## 3      ddimer log_leukocyte 0.8221853 0.7950422 0.02714305 0.04705981 0.5767779
```

```
##        p-value
## 1 3.281860e-06
## 2 3.341962e-03
## 3 5.640895e-01
##
## Confusion matrix:
##           Outcome +    Outcome -      Total
## Test +           75           20         95
## Test -            8           67         75
## Total            83           87        170
##
## Point estimates and 95% CIs:
## -------------------------------------------------------------
## Apparent prevalence *                  0.56 (0.48, 0.63)
## True prevalence *                      0.49 (0.41, 0.57)
## Sensitivity *                          0.90 (0.82, 0.96)
## Specificity *                          0.77 (0.67, 0.85)
## Positive predictive value *            0.79 (0.69, 0.87)
## Negative predictive value *            0.89 (0.80, 0.95)
## Positive likelihood ratio              3.93 (2.66, 5.81)
## Negative likelihood ratio              0.13 (0.06, 0.24)
## False T+ proportion for true D- *      0.23 (0.15, 0.33)
## False T- proportion for true D+ *      0.10 (0.04, 0.18)
## False T+ proportion for T+ *           0.21 (0.13, 0.31)
## False T- proportion for T- *           0.11 (0.05, 0.20)
## Correctly classified proportion *      0.84 (0.77, 0.89)
## -------------------------------------------------------------
## * Exact CIs
```

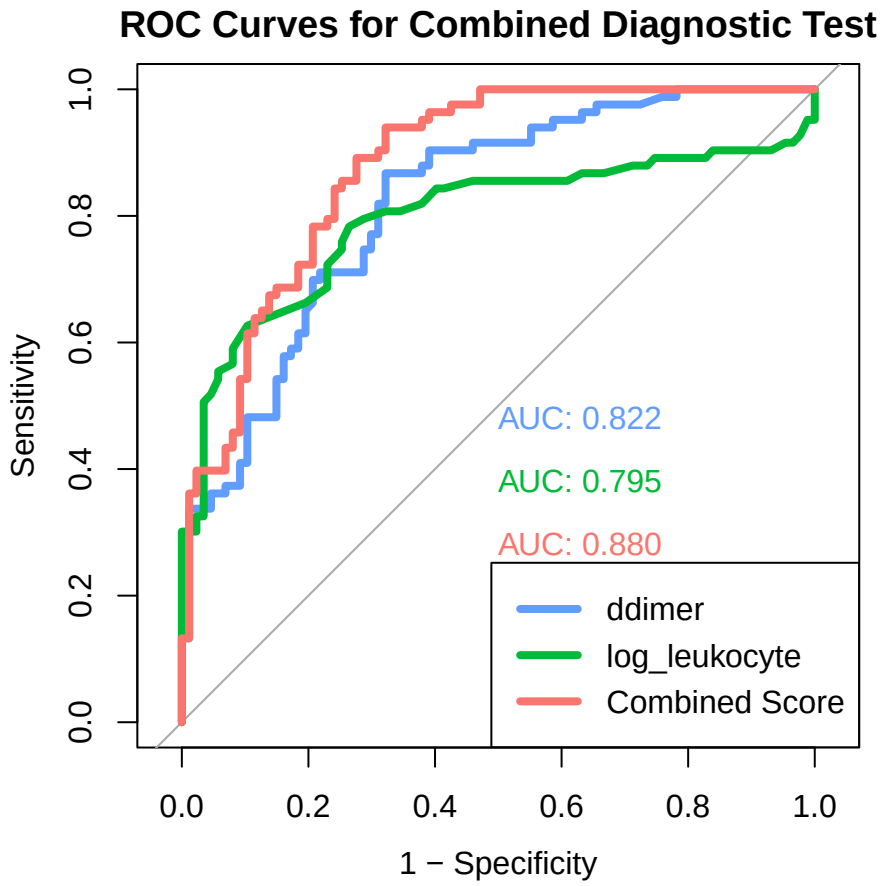
```
#mlComb Function
fit.ml <- mlComb(markers = markers,
                     status = status,
                     event = "needed",
                     method = "knn",
                     resample = "repeatedcv", nfolds = 10, nrepeats = 5,
                     preProcess = c("center", "scale"),
                     direction = "<", cutoff.method = "youden")
```




```
## k-Nearest Neighbors
##
## 170 samples
##   2 predictor
##   2 classes: 'not_needed', 'needed'
##
## Pre-processing: centered (2), scaled (2)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 153, 153, 153, 153, 152, 153, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.7446814  0.4887963
##   7  0.7344853  0.4684145
##   9  0.7332435  0.4664939
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
##
##
## Area Under the Curves of markers and combined score:
##                      AUC     SE.AUC LowerLimit UpperLimit         z
## ddimer        0.8221853 0.03136528  0.7607105  0.8836601 10.272036
## log_leukocyte 0.7950422 0.03718164  0.7221676  0.8679169  7.935159
## Combined      0.9392051 0.01560800  0.9086140  0.9697962 28.139738
```

```
##                     p.value
## ddimer         9.419585e-25
## log_leukocyte  2.102258e-15
## Combined      3.199976e-174
##
## Area Under the Curve comparison of markers and combined score:
##   Marker1 (A)   Marker2 (B)   AUC (A)   AUC (B)      |A-B|  SE(|A-B|)         z
## 1    Combined        ddimer 0.9392051 0.8221853 0.11701980 0.02631726 4.4465034
## 2    Combined log_leukocyte 0.9392051 0.7950422 0.14416286 0.03412920 4.2240329
## 3      ddimer log_leukocyte 0.8221853 0.7950422 0.02714305 0.04705981 0.5767779
##        p-value
## 1 8.727931e-06
## 2 2.399690e-05
## 3 5.640895e-01
##
## Confusion matrix:
##           Outcome +    Outcome -      Total
## Test +           70           11         81
## Test -           13           76         89
## Total            83           87        170
##
## Point estimates and 95% CIs:
## -------------------------------------------------------------
## Apparent prevalence *                 0.48 (0.40, 0.55)
## True prevalence *                     0.49 (0.41, 0.57)
## Sensitivity *                         0.84 (0.75, 0.91)
## Specificity *                         0.87 (0.79, 0.94)
## Positive predictive value *           0.86 (0.77, 0.93)
## Negative predictive value *           0.85 (0.76, 0.92)
## Positive likelihood ratio             6.67 (3.81, 11.68)
## Negative likelihood ratio             0.18 (0.11, 0.30)
## False T+ proportion for true D- *     0.13 (0.06, 0.21)
## False T- proportion for true D+ *     0.16 (0.09, 0.25)
## False T+ proportion for T+ *          0.14 (0.07, 0.23)
## False T- proportion for T- *          0.15 (0.08, 0.24)
## Correctly classified proportion *     0.86 (0.80, 0.91)
## -------------------------------------------------------------
## * Exact CIs
```

```r
#mathComb Function
fit.math <- mathComb(markers = markers,
                         status = status,
                         event = "needed",
                         method = "distance",
                         distance = "euclidean",
                         direction = "<",
                         cutoff.method = "youden")
```




```
## Method: distance
## Distance: euclidean
## Samples: 170
## Markers: 2
## Events: not_needed, needed
## Standardization: none
## Transform: none
##
##  Kappa       Accuracy
##  0.6140085   0.8058824
##
## Area Under the Curves of markers and combined score:
##                     AUC     SE.AUC LowerLimit UpperLimit         z      p.value
## ddimer        0.8221853 0.03136528  0.7607105  0.8836601 10.272036 9.419585e-25
## log_leukocyte 0.7950422 0.03718164  0.7221676  0.8679169  7.935159 2.102258e-15
## Combined      0.8797950 0.02536950  0.8300717  0.9295183 14.970537 1.143941e-50
##
## Area Under the Curve comparison of markers and combined score:
##   Marker1 (A)   Marker2 (B)   AUC (A)   AUC (B)      |A-B|  SE(|A-B|)         z
## 1    Combined        ddimer 0.8797950 0.8221853 0.05760975 0.01359213 4.2384625
## 2    Combined log_leukocyte 0.8797950 0.7950422 0.08475280 0.03874061 2.1876993
## 3      ddimer log_leukocyte 0.8221853 0.7950422 0.02714305 0.04705981 0.5767779
##        p-value
## 1 2.250558e-05
## 2 2.869151e-02
```

```
## 3 5.640895e-01
##
## Confusion matrix:
##           Outcome +    Outcome -      Total
## Test +           78           28        106
## Test -            5           59         64
## Total            83           87        170
##
## Point estimates and 95% CIs:
## -------------------------------------------------------------
## Apparent prevalence *                 0.62 (0.55, 0.70)
## True prevalence *                     0.49 (0.41, 0.57)
## Sensitivity *                         0.94 (0.86, 0.98)
## Specificity *                         0.68 (0.57, 0.77)
## Positive predictive value *           0.74 (0.64, 0.82)
## Negative predictive value *           0.92 (0.83, 0.97)
## Positive likelihood ratio             2.92 (2.14, 3.98)
## Negative likelihood ratio             0.09 (0.04, 0.21)
## False T+ proportion for true D- *     0.32 (0.23, 0.43)
## False T- proportion for true D+ *     0.06 (0.02, 0.14)
## False T+ proportion for T+ *          0.26 (0.18, 0.36)
## False T- proportion for T- *          0.08 (0.03, 0.17)
## Correctly classified proportion *     0.81 (0.74, 0.86)
## -------------------------------------------------------------
## * Exact CIs
```

8 Comparing the performance of classifiers

As seen in Table 1, the results of the four models set up, the combined diagnostic tests have performed higher than the single diagnostic tests. Among the models that have been built, the K-Nearest Neighbors `knn` method, one of the machine learning algorithms, returns the highest AUC value. The (`knn`) method parameters that gives the best results will be used in the next step to make predictions on the test data.

Table 1: Combination results for train data

Metot	AUC	Accuracy
ddimer	0.822	0.771
log(leukocyte)	0.795	0.765
scoring	0.878	0.824
lassoreg	0.908	0.835
knn	0.939	0.745
distance(euclidean)	0.880	0.806

9 Predicting the class labels of test samples

The class labels of the test cases are estimated based on the parameters of the model trained, e.g. If we are training a model using the `knn` method, one of the machine learning algorithms, the labels of the test set are estimated by the parameters we obtained from the training set. However, the important point is that the test set must have gone through the same standardization stages as the training set. The `predComb` function makes predictions in line with the training model by applying this standardization process in the test set. The test data set is created from the exampleData1 sample data as follows, and the `predComb` function is applied to this data set.

```
testData <- exampleData1[-inTrain, -1]
```

```
predComb(fit.nonlin, testData)
```

```
##      comb.score     labels
## 1  1.000000e+00     needed
## 2  9.339987e-01     needed
## 3  1.000000e+00     needed
## 4  1.479751e-16 not_needed
## 5  1.000000e+00     needed
## 6  1.041296e-09 not_needed
## 7  1.000000e+00     needed
## 8  1.000000e+00     needed
## 9  6.245920e-13 not_needed
## 10 1.730838e-27 not_needed
## 11 9.477324e-07 not_needed
## 12 1.266500e-33 not_needed
## 13 8.276221e-33 not_needed
## 14 1.213456e-31 not_needed
## 15 7.402595e-27 not_needed
## 16 1.000000e+00     needed
## 17 1.082540e-11 not_needed
## 18 1.000000e+00     needed
## 19 2.920516e-29 not_needed
## 20 1.000000e+00     needed
## 21 3.454558e-28 not_needed
## 22 7.030873e-29 not_needed
## 23 5.895762e-28 not_needed
## 24 1.000000e+00     needed
## 25 7.190401e-12 not_needed
## 26 6.749013e-29 not_needed
## 27 4.725644e-25 not_needed
## 28 1.719394e-33 not_needed
## 29 2.321893e-24 not_needed
## 30 1.471633e-25 not_needed
## 31 2.944632e-30 not_needed
## 32 5.051824e-32 not_needed
## 33 3.122348e-34 not_needed
## 34 3.093785e-36 not_needed
## 35 5.574296e-35 not_needed
## 36 7.743622e-35 not_needed
## 37 2.474333e-36 not_needed
## 38 2.099961e-35 not_needed
## 39 7.600877e-27 not_needed
## 40 1.302778e-32 not_needed
## 41 3.027438e-39 not_needed
## 42 3.189182e-37 not_needed
## 43 9.858140e-28 not_needed
## 44 1.161957e-27 not_needed
## 45 2.581951e-28 not_needed
## 46 1.000000e+00     needed
## 47 7.513786e-27 not_needed
## 48 8.534483e-30 not_needed
## 49 1.147069e-29 not_needed
```

```
## 50 2.731496e-27 not_needed
## 51 1.324388e-36 not_needed
## 52 1.135819e-24 not_needed
## 53 4.267400e-27 not_needed
## 54 2.778275e-36 not_needed
## 55 9.237483e-38 not_needed
```

The `predComb` function returns the combined score of the applied method and the estimated label for the models trained using the `linComb`, `nonlinComb`, or `mathComb` functions. For the models trained using `mlComb` function, the probability of positive and negative cases for each test observation is given as output.

10 Session info

```
sessionInfo()

## R version 4.2.0 Patched (2022-06-07 r82464 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19044)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United Kingdom.utf8
## [2] LC_CTYPE=English_United Kingdom.utf8
## [3] LC_MONETARY=English_United Kingdom.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United Kingdom.utf8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] caret_6.0-92    lattice_0.20-45 ggplot2_3.3.6   dtComb_0.99.6
## [5] knitr_1.39.2
##
## loaded via a namespace (and not attached):
##  [1] splines_4.2.0        foreach_1.5.2        prodlim_2019.11.13
##  [4] epiR_2.0.46          highr_0.9            stats4_4.2.0
##  [7] pander_0.6.5         globals_0.15.0       gdtools_0.2.4
## [10] ipred_0.9-12         pillar_1.7.0         glue_1.6.2
## [13] pROC_1.18.0          uuid_1.1-0           digest_0.6.29
## [16] hardhat_0.2.0        colorspace_2.0-3     recipes_0.2.0
## [19] htmltools_0.5.2      Matrix_1.4-1         plyr_1.8.7
## [22] timeDate_3043.102    pkgconfig_2.0.3      listenv_0.8.0
## [25] purrr_0.3.4          scales_1.2.0         gower_1.0.0
## [28] officer_0.4.2        lava_1.6.10          tibble_3.1.7
## [31] proxy_0.4-26         generics_0.1.2       ellipsis_0.3.2
## [34] withr_2.5.0          nnet_7.3-17          cli_3.3.0
## [37] survival_3.3-1       magrittr_2.0.3       crayon_1.5.1
## [40] evaluate_0.15        future_1.25.0        fansi_1.0.3
## [43] parallelly_1.31.1    nlme_3.1-157         MASS_7.3-57
## [46] xml2_1.3.3           class_7.3-20         tools_4.2.0
## [49] data.table_1.14.2    lifecycle_1.0.1      stringr_1.4.0
```

```
## [52] flextable_0.7.0      glmnet_4.1-4         munsell_0.5.0
## [55] zip_2.2.0            compiler_4.2.0       e1071_1.7-9
## [58] systemfonts_1.0.4    rlang_1.0.2          classInt_0.4-3
## [61] units_0.8-0          grid_4.2.0           iterators_1.0.14
## [64] rstudioapi_0.13      rmarkdown_2.14       base64enc_0.1-3
## [67] gtable_0.3.0         ModelMetrics_1.2.2.2 codetools_0.2-18
## [70] DBI_1.1.2            reshape2_1.4.4       R6_2.5.1
## [73] zoo_1.8-10           lubridate_1.8.0      dplyr_1.0.9
## [76] fastmap_1.1.0        future.apply_1.9.0   utf8_1.2.2
## [79] shape_1.4.6          KernSmooth_2.23-20   stringi_1.7.6
## [82] parallel_4.2.0       BiasedUrn_1.07       Rcpp_1.0.8.3
## [85] vctrs_0.4.1          sf_1.0-7             rpart_4.1.16
## [88] tidyselect_1.1.2     xfun_0.31
```

References

[1] Hizir Yakup Akyildiz, Erdogan Sozuer, Alper Akcan, Can Kuçuk, Tarik Artis, İsmail Biri, Namık Yılmaz, et al. The value of d-dimer test in the diagnosis of patients with nontraumatic acute abdomen. *Turkish Journal of Trauma and Emergency Surgery*, 16(1):22–26, 2010.

[2] Cristóbal León, Sergio Ruiz-Santana, Pedro Saavedra, Benito Almirante, Juan Nolla-Salas, Francisco Álvarez-Lerma, José Garnacho-Montero, María Ángeles León, EPCAN Study Group, et al. A bedside scoring system ("candida score") for early antifungal treatment in nonneutropenic critically ill patients with candida colonization. *Critical care medicine*, 34(3):730–737, 2006.

[3] John Q Su and Jun S Liu. Linear combinations of multiple diagnostic markers. *Journal of the American Statistical Association*, 88(424):1350–1355, 1993.

[4] Strother H Walker and David B Duncan. Estimation of the probability of an event as a function of several independent variables. *Biometrika*, 54(1-2):167–179, 1967.

[5] Chunling Liu, Aiyi Liu, and Susan Halabi. A min–max combination of biomarkers to improve diagnostic accuracy. *Statistics in medicine*, 30(16):2005–2014, 2011.

[6] Margaret Sullivan Pepe and Mary Lou Thompson. Combining diagnostic test results to increase accuracy. *Biostatistics*, 1(2):123–140, 2000.

[7] Margaret Sullivan Pepe, Tianxi Cai, and Gary Longton. Combining predictors for classification using the area under the receiver operating characteristic curve. *Biometrics*, 62(1):221–229, 2006.

[8] G Sameera, R Vishnu Vardhan, and KVS Sarma. Binary classification using multivariate receiver operating characteristic curve for continuous data. *Journal of biopharmaceutical statistics*, 26(3): 421–431, 2016.

[9] Nicolae Todor, Irina Todor, and Gavril Săplăcan. Tools to identify linear combination of prognostic factors which maximizes area under receiver operator curve. *Journal of clinical bioinformatics*, 4(1): 1–7, 2014.

[10] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

[11] R Core Team et al. R: A language and environment for statistical computing. 2013.

[12] Trevor Hastie. gam: Generalized additive models. r package version 1.16. 1. *Von https://CRAN. R-project. org/package= gam abgerufen*, 2019.

[13] Hajk-Georg Drost. Philentropy: information theory and distance quantification with r. *Journal of Open Source Software*, 3(26):765, 2018.

[14] Max Kuhn. Caret: classification and regression training. *Astrophysics Source Code Library*, pages ascl–1505, 2015.

[15] Xavier Robin, Natacha Turck, Alexandre Hainard, Natalia Tiberti, Frédérique Lisacek, Jean-Charles Sanchez, and Markus Müller. proc: an open-source package for r and s+ to analyze and compare roc curves. *BMC bioinformatics*, 12(1):1–8, 2011.