ROC632: An Overview

Catarina Santos[1] and Ana Cristina Braga[2]

[1] University of Minho, Portugal
cfssantos_13@hotmail.com
[2] Algoritmi Centre, University of Minho, Portugal
acb@dps.uminho.pt

Abstract. The present paper aims to analyze and explore the ROC632 package, specifying its main characteristics and functions. More specifically, the goal of this study is the evaluation of the effectiveness of the package and its strengths and weaknesses. This package was created in order to overcome the lack of information concerning incomplete time-to-event data, adapting the 0.632+ bootstrap estimator for the evaluation of time dependent ROC curves. By applying this package to a specific dataset (DLBCLpatients), it becomes possible to assess tangible data, determining if it is able to analyze complete and incomplete data efficiently and without bias.

Keywords: ROC632 package, 0.632+ bootstrap, ROC curves

1 Introduction

The ROC632 package is a R package currently available in version 0.6. It was created by Yohann Foucher and it was first published in December 27th of 2013 [5] on CRAN repository.

This package was created in order to overcome the lack of information concerning incomplete time-to-event data (in this case, patient death [5]), adapting the 0.632+ bootstrap estimator for the evaluation of time dependent ROC curves, where the results do not depend on the incidence of the event [6].

This package allows estimation of prognostic capacity of microarray data and it relies on four main functions: *ROC, AUC, boot.ROC* and *boot.ROCt* [5,6].

The information listed above was adapted from [5] and [6] and further details about the features in which this package is based can be found in sections 1.1, 1.2 and 1.3.

1.1 ROC Curves

A ROC curve is a plot used to evaluate the relationship between sensitivity and one minus specificity (*false positive rate* (FPR)) [2,4,9].

Thus, *sensitivity*, or *true positive rate*, is the proportion of true positives (TP), i.e., the correctly classified positives divided by the true positives plus those who should be classified as such (FN) [2–4,9,10]. In medical terms, sensitivity can also be perceived as the ability to identify diseased patients from a given sample [2].

Similarly, *specificity*, or *true negative rate*, is the proportion of correctly classified negatives (true negatives - TN) from all the expected negatives [2–4,9,10], which can also be interpreted as the capacity to disregard all healthy individuals from a given sample [2]. Specificity can be formulated as $TN/(TN + FP)$, from which we can obtain the false positive rate: $FPR = 1 - specificity$.

Accuracy, which is a key parameter for any test, [2–4, 13] is the number of correctly classified objects out of all the given objects [13], i.e., the proportion of true positives and true negatives (correctly classified elements) in a sample [2–4], which implies that test accuracy is measured by sensitivity and specificity [2].

1.2 Area Under the ROC Curve

The most common and most important index able to attain the essential features of a ROC curve is the Area Under the ROC Curve (AUC) [4,9,10], which reduces it to a single scalar value [3, 10]. The AUC can be computed by the trapezoidal method [3, 9, 10].

The value of the AUC ranges from 0 to 1, because it belongs to the *unit square* [3, 4]. The ideal value for the AUC is 1 [2], meaning that every positive scored higher that every negative [4]; inversely, an AUC of 0 means that every negative scored higher than every positive and that the test has no accuracy [2,4] and thus should be discarded. Despite starting at 0, one should only consider AUC values ranging from 0.5 to 1, because the diagonal line (see section 1.1) has an area of 0.5 [3].

Accuracy can also be estimated through the AUC: a test with an AUC value below 0.5 has *no accuracy*, between 0.5 and 0.7 has *low accuracy*, ranging from 0.7 to 0.9 has *moderate accuracy* and above 0.9 has *high accuracy*, emphasizing that "the greater the AUC, the better the test" [2, 3].

1.3 The Bootstrap Method

Finding a method for validating predictive models and obtaining an unbiased performance has been a target of discussion by multiple authors [11,16]. Although there are several approaches to estimate the error rate of a prediction rule, such as the jackknife (leave-one-out) method and cross-validation, the bootstrap method has been considered the most efficient throughout the years, as it is capable of directly assessing the variability, returns higher accuracy and it is able to calculate the variance of a point estimate of prediction error [1, 14, 15].

The *bootstrap method* separates the available data into two sets: the *training set*, which is used to obtain a predictive model and the *test set*, used to evaluate its performance [4,5,8,16]. It draws random instances with replacement (resampling) from the original dataset, which means that some sets can be used multiple times and some might not be used at all, although, typically, eventually all the data will be selected at least once [10, 14, 15].

The *0.632 bootstrap estimator*, or *0.632 bootstrap resampling variable*, evaluates independent data, estimated on a per-subject basis [1, 14].

However, since this method can still be biased, another estimator, the *0.632+ estimator*, was created to improve it [1, 14, 15].

2 Materials and Methods

This paper focused mainly on the exploration of the ROC632 package, using the *DLBCL* dataset, created by Rosenwald et al. [12]. The *DLBCLpatients* dataset and the DLBCLgenes matrix concern, respectively, 240 patients affected by a diffuse large b-cell lymphoma (DLBCL) treated with anthracycline based therapy and their clinical information, based on the scientific discoveries made by Rosenwald [12].

The assessment of the *boot.ROC* function, which builds a model relying on logistic regression with lasso penalty and deals only with complete data, consisted on varying parameters, that is, assigning different values to its arguments, namely the lasso penalty (*lambda1* – tuning parameter) and/or the number of bootstrap iterations (*N.boot*) and, in the cases where lambda was *NULL*, the fold for cross-validation (*fold.cv*). Then, the significance of the difference between the *apparent*, *cross-validation*, *0.632* and *0.632+* bootstrap curves for each condition was estimated. The study of the *boot.ROCt* function, which is capable of dealing with censored data and draws a model by applying the Cox model with lasso penalty, included the process listed for the first function and the modification of the prognostic limit for which the variable is evaluated (*pro.time* argument).

Both functions return a vector (*Coef*) with the regression coefficients obtained in the logistic or Cox model with lasso penalty (*boot.ROC* and *boot.ROCt*, respectively). These coefficients were used to determine the significant features for each result (those whose coefficient is nonzero), which could be related to the emergence of this type of lymphoma.

Since the ROC632 package does not calculate the standard error of the AUC, three functions were added to enable comparison between them. The first function calculates the *standard error* of a given curve, the second function determines the *z score* of two compared curves and the third function estimates the *p value* of the difference between the curves.

```r
"st_error" = function (A) {
Q1 = A / (2 - A)
Q2 = (2 * A ^2) / (1 + A)
sqrt((A*(1-A) + (ndead-1) * (Q1-A^2) + (nalive-1) * (Q2-A^2))/(ndead*nalive))
}
"z" = function(A1, A2, se1, se2) (A1-A2)/sqrt(se1^2 + se2^2)
"pval" = function(z) {
if (z < 0) p = 2*pnorm(z,lower.tail = T )
else p = 2*pnorm(z,lower.tail = F)
return (p)
}
```

R Script 1.1. Standard Error; Z Score and P value Functions

3 Results and Discussion

3.1 Evaluation of the *boot.ROC* function

As mentioned in section 2, the *boot.ROC* function constructs a lasso penalized model for complete data according to a scoring system using logistic regression and estimates the resulting traditional four curves: apparent, cross-validation, 0.632 and 0.632+. The produced results depend on patient survival.

The boot.ROC function returns a list with 10 elements. *Coef* is a replica of *Model@penalized*, that is, the regression coefficients for the penalized co-variables (features and status). *Signature* is the score for each patient, obtained by the sum of the regression (*Coef*) multiplied by the values of the features. *Lambda* is the value of the lasso penalty and AUC is the mean of the area under the curve for the four estimators. This function also generates 4 important data frames for the false positive and false negative rates (obtained from the ROC function's argument *cut.values*): *ROC.Apparent* for the apparent estimator, *ROC.CV* for the cross-validation estimator, *ROC.632* for the 0.632 bootstrap estimator and *ROC.632p* for the 0.632+ bootstrap estimator (these data frames generate each point of the traditional curve). The last element is *Model*, which is a penfit object with 15 elements (see reference [8] for details).

The evaluation of this function consisted of varying the values of its arguments, specifically assessing its performance by assigning different values to the lasso penalty (*lambda1*), ranging from three fixed values – 10, 15 and 20 – to variable lambda values (*lambda1* = NULL) while increasing the number of bootstrap iterations (*N.boot*) from 2, 50 and 100 to 1000. The *precision* argument (generates the points for each curve) remained constant throughout the process, set as a vector of 4 values (0.05, 0.35, 0.65 and 0.95).

Fixed Lambda Values. It was verified that there were no significant alterations in the curves by altering the number of bootstrap iterations within the same tuning parameter (lambda). As such, only the two extreme values (2 and 1000 iterations) were included in Figure 1. This similarity, however, was not verified for the different lasso penalties. For 2 bootstrap iterations, there were significant fluctuations between the apparent curves for a lasso penalty of 10 and 15 ($p < 0.002$) and 10 and 20 ($p < 0.0003$). For 1000 bootstrap iterations, there were also significant changes for the apparent curves between the 10 and 20 lasso penalties ($p < 0.01$).

For the three fixed lambdas, it was found that increasing the lasso penalty decreased the range of values present in the signature, that regardless of the tuning parameter more patients were assigned a negative than a positive score and that the signature was invariable with the increasing bootstrap iterations within the same lambda.

The significant features were identified by matching the unique ID in Rosenwald's NEJM_13Web _13Fig1data dataset (available at `http://llmpp.nih.gov/DLBCL`) to the nonzero coefficients obtained in each result. For lasso penalties of 10, 15 and 20, 42, 24 and 10 significant features were found, respectively. For the latter case, according to the NCBI database and publications [7], the 10 features are well-known overexpression transcriptional factors, genes or cell types related to the diffuse large-b-cell or other similar types of lymphoma, while the results for the other lambdas included hypothetical proteins and other non-cancer related genes. These results indicate that increasing the lasso penalty renders higher accuracy in highlighting risk factors (high influence on patient survival).

Lastly, it was ascertained that the number of bootstrap iterations had no influence in the level of fit (log likelihood) of the produced model. However, this level was directly proportional to the lasso penalty (tuning parameter of 20 produced the best results), which implies that a higher lasso penalty could be linked to a higher accuracy in determining risk factors and patient survival.

Variable Lambda Values. In this subsection, the *boot.ROC* function's argument *lambda1* was set to NULL, which means that the value for the lasso penalty is generated by cross-validation by re-estimating the tuning parameter and selecting features at each bootstrap iteration. The fold for cross-validation was set to 5 (default), 10 and 20 and the number of bootstrap iterations was increased from 2 to 10.

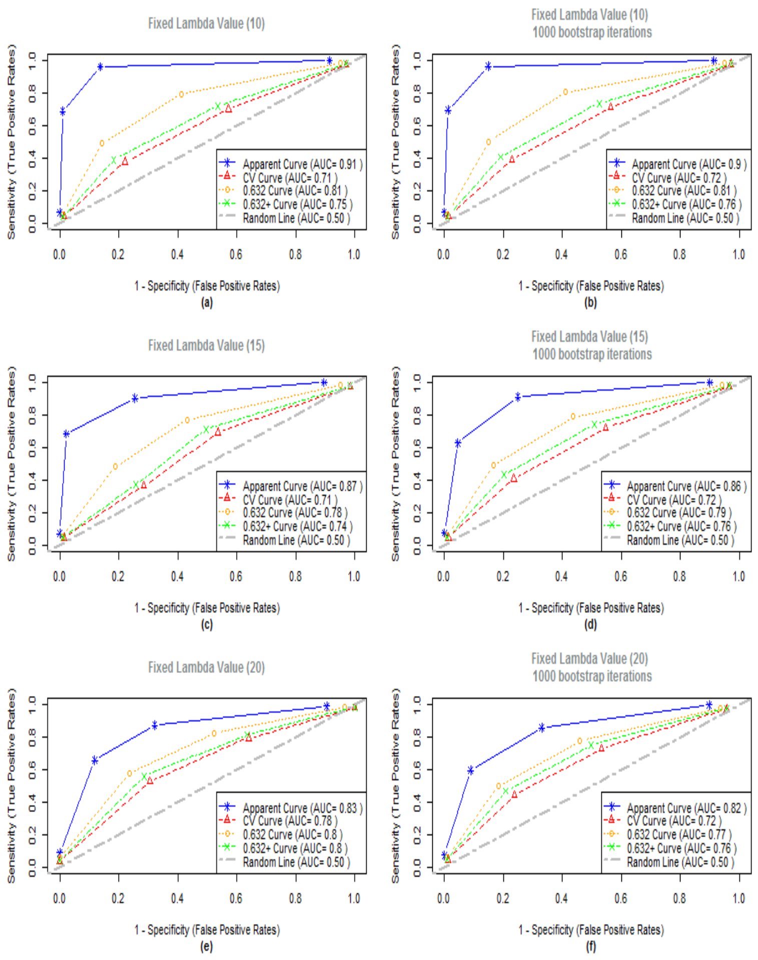

Fig. 1. ROC curves for a fixed tuning parameter of 10 (*boot.ROC* function).

Figure 2(a) demonstrates the results with higher test accuracy (higher AUC values) attained for the variable lasso penalty, found for the 5-fold at 2 bootstrap iterations (with a lesser overestimation of the apparent curve). Figure 2(b) shows the calculated signature for this fold, with values ranging from -1.008426 to 0.5911469 (narrower range than the observed for the fixed lambdas).

The level of fit of the model (*Log likelihood*) had increasingly negative values for the 20, 10 and 5 cross-validation folds, which implies that there is better adjustment for smaller folds.

The significant feature search for the variable lambdas yielded highly specific matches, revealing features that are all characteristic of the diffuse large b-cell lymphoma signature [7, 11, 12] and a better performance than the observed in section 3.1.1. For 2 and 10 bootstrap iterations, 4 significant features were acquired, from which three are coincident with the ones found for the fixed 20 lasso penalty.

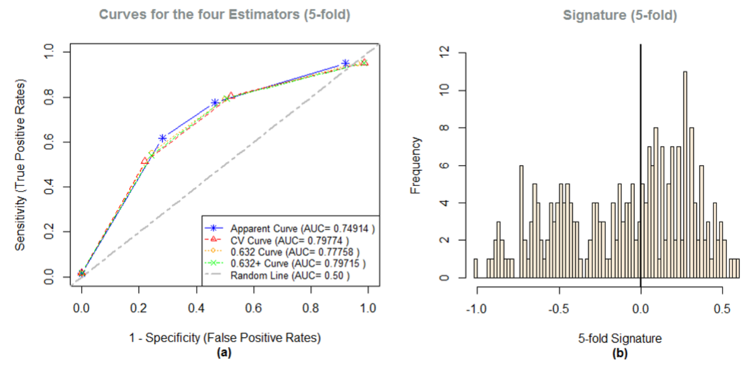

Fig. 2. Produced curves and signature for a variable tuning parameter (5-fold).

3.2 Assessment of the *boot.ROCt* function

The *boot.ROCt* function constructs a lasso penalized model using the Cox's proportional hazards model, given a calculated signature (see section 2 for details) and estimates the corresponding time-dependent curve. Since this function is capable of dealing with censored data, the argument *status* from the *boot.ROC* function is replaced by the binary argument *failure*, where 0 means right-censoring and 1 implies the event (in this case, death) took place. Right-censoring, for the newDLBCLpatients dataset, since the maximum follow up time is 21.8 years, is perceived as the patient leaving the study before that time period or surviving after it [5, 6, 10].

This function returns a list similar to the one described in subsection 3.1. The assessment of the *boot.ROCt* function included the process listed for the *boot.ROC* function, while varying the maximum prognostic time (*pro.time*) for which each variable is evaluated (from 1 to 16 years) and changing the lasso penalty (*lambda1*) from variable to fixed (15). The proportion of nearest neighbors (*prop*) was kept constant at a value of 0.02 during the course of the study and only 2 bootstrap iterations (*N.boot*) were considered.

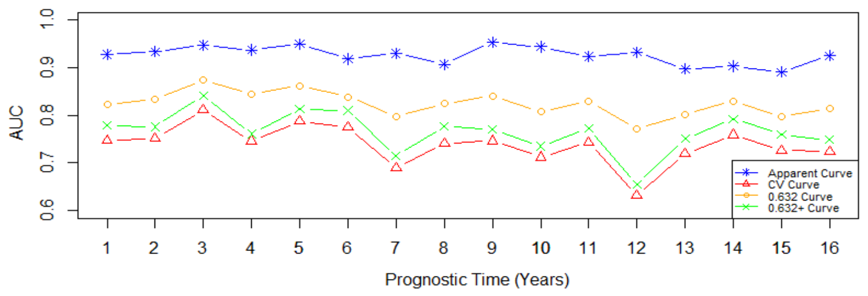

Fig. 3. Time dependent ROC curves for variable **(a)** and fixed **(b)** tuning parameters.

Figure 3 illustrates the performance of the four estimators for right-censored data (*Apparent*, *Cross-Validation*, *0.632* and *0.632+*) by varying the argument *pro.time* from 1 to 16 years, according to a variable lasso penalty.

Using the functions described in section 2, there were significant differences ($p < 0.05$) between the curves for fixed and variable penalties: in the apparent curve, for prognostic times of 9 and 14 years, in the cross-validation curve and in the 0.632+ bootstrap curve, for 3 and 12 years. These differences are due to the small number of observations for those specific years, leading the model to under- or overestimate those values depending on the used approach.

The most negative log likelihood (-812.4193) and lowest number of significant features (which ranged from 4 to 32) for the variable penalty were found for a prognostic time of eleven years, which means that those particular models were the most accurate in predicting patient outcome. For the fixed lambda value of 15, independently of the prognostic time, a log likelihood of -739.6779 was achieved and 68 relevant genes were found, indicating low reliability.

4 Conclusions

From the results described throughout section 3, the strengths and weaknesses of the ROC632 package could be highlighted and, independently of the condition or the function in use, some patterns in the outcomes were identified.

The most significant disadvantages of this package are that it does not calculate the standard error for the estimated curves, forcing the user to calculate it using an additional method and that the number of patients assigned to the training and test sets is not explicitly shown in the results.

The apparent curve seemed to be overly optimistic and the cross-validation curve considerably pessimistic. These results were expected, since these curves only represent the training and test sets, respectively. The 0.632 and 0.632+ bootstrap curves had an overall similar performance, with marginally lower values for latter, since the 0.632+ estimator's performance depends on the amount of overfitting, whereas the former has a constant weight [1, 14, 15]. Hence, the 0.632+ estimator provided the best results with the least variance (similar results within the same condition – fixed or variable lambda) and bias (no over- or underestimation) and it should thus be used in future analysis.

Finally, the signature created was able to create an overall efficient prognosis for up to 10 years, being capable of attributing a higher score to patients who survived the longest (and were still alive). However, since most patients had died after that time point (only 26 patients had survived), the predictions for longer timeframes were considerably erroneous, with patients who had survived having the same score as patients who did not. Hence, although this scoring system could be highly accurate for well documented data, with as many observations possible, it isn't advised for small datasets where the data is overly repetitive in some cases and missing in others and the number of examples is limited.

References

1. Ambroise, C., McLachlan, G.J.: Selection bias in gene extraction on the basis of microarray gene-expression data. Proceedings of the National Academy of Sciences 99(10), 6562–6566 (2002)
2. Collinson, P.: Of bombers, radiologists and cardiologists: time to ROC. Heart (1998)
3. Fawcett, T.: An introduction to ROC analysis. Pattern Recognition Letters 27(8), 861–874 (2006)
4. Flack, P.: ROC Analysis. Encyclopedia of Machine Learning, pp. 869–874. Springer Publishing Company, 1 edn. (2011)
5. Foucher, Y.: ROC632: Construction of diagnostic or prognostic scoring system and internal validation of its discriminative capacities based on ROC curve and 0.633+ bootstrap resampling. (2013), R package version 0.6 `https://cran.r-project.org/web/packages/ROC632/index.html`
6. Foucher, Y., Danger, R.: Time dependent ROC curves for the estimation of true prognostic capacity of microarray data. Statistical Applications in Genetics and Molecular Biology 11(6) (2012)
7. Geer, L.Y., Marchler-Bauer, A., Geer, R.C., et al.: The NCBI BioSystems database. Nucleic Acids Research 38(Database), D492–D496 (2009)
8. Goeman, J., Meijer, R., Chaturvedi, N.: L1 (Lasso and Fused Lasso) and L2 (Ridge) Penalized Estimation (2013), R package version 0.9-47 `https://cran.r-project.org/web/packages/penalized/index.html`
9. Gonen, M.: Receiver Operating Characteristic (ROC) Curves (Paper 210-31). SUGI 31 Proceedings, pp. 1–18. SAS Institute Inc (2006)
10. Krzanowski, W.J., Hand, D.J.: ROC curves for continuous data. CRC Press (2009)
11. Liu, H., Li, J., Wong, L.: Use of extreme patient samples for outcome prediction from gene expression data. Bioinformatics 21(16), 3377–3384 (2005)
12. Rosenwald, A., Wright, G., Chan, W.C.e.a.: The use of molecular profiling to predict survival after chemotherapy for diffuse large-b-cell lymphoma. New England Journal of Medicine 346(25), 1937–1947 (2002)
13. Sammut, C., Webb, G.: Encyclopedia of Machine Learning. Springer Publishing Company (2011)
14. Steyerberg, E.W., Harrell, F.E., Borsboom, G.J., et al.: Internal validation of predictive models: Efficiency of some procedures for logistic regression analysis. Journal of Clinical Epidemiology 54(8), 774–781 (2001)
15. Vu, T., Sima, C., Braga-Neto, U.M., Dougherty, E.R.: Unbiased bootstrap error estimation for linear discriminant analysis. J Bioinform Sys Biology 2014(1) (2014)