Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

ROC and AUC in R with a Binary Predictor

John Muschelli
Department of Biostatistics, Johns Hopkins Bloomberg School of Public Health

Abstract

The abstract of the article.

Keywords: roc, auc, area under the curve, R.

1. Introduction

In many applications, receiver operator characteristic (ROC) curves are used to show how a predictor compares to the true outcome. One of the large draws of th ROC curve is that it is threshold-agnostic; it shows the performace of a predictor without a specific threshold and also gives a criteria to choose an optimal threshold based on a certain cost function.

Many predictors, especially medical tests, result in a binary decision; a value is higher than a pre-determined threshold or not. Similarly, some are simply categorical or discrete; for example, blood pressure may be low, normal, or high. These are useful indicators of presence disease, which is a primary outcome of interest.

Fawcett (2006) describes (in Fig. 6) how ties are handled in a predictor. Ties are distinctly relevant for discrete and binary predictors or models that predict a discrete number of values, where many observations can have the same value/risk score. When drawing the ROC curve, one can assume that all the ties do not correctly classify the outcome (Fawcett called the "pessimistic" approach) or that all the ties do correclty classify the outcome (called the "optimistic" approach). But Fawcett notes: > Any mixed ordering of the instances will give a different set of step segments within the rectangle formed by these two extremes. However, the ROC curve should represent the *expected* performance of the classifier, which, lacking any other information, is the average of the pessimistic and optimistic segments.

Therefore,

With some assumptions, you can think of the binary predictor as generating from a distribution of values from a continuous distribution that has been thresholded. Therefore, the

sensitivity of this thresholded predictor actually represents one point on the ROC curve of the true underlying continuous distribution. Therefore the ROC curve of a binary predictor is not really appropriate. But alas, ROC and AUC analysis is done on binary predictors and used to inform if one variable is more predictive than the other.

A more appropriate comparison of a continuous predictor and the binary predictor may be to compare the sensitivity and specificity (or overall accuracy) of the continuous predictor given the optimal threshold versus that of the binary predictor.

This expectation directly applies to the assignment of a half probability of success when the data are tied. This half probability is linked to how ties are treated in the Wilcoxon rank sum test. As much of the theory of ROC curve testing, and therefore testing of differences in AUC, is based on the theory of the Wilcoxon rank-sum test, this treatment of ties is relevant to statistical inference.

```
R> library(cranlogs)
R> library(ggplot2)
R> library(dplyr)
```

In this tutorial, we will show the calculation of the AUC

2. Simple Example

Here we will create a simple scenario where there is a binary predictor X and a binary outcome Y.

```
R> x = c(rep(0, 52), rep(1, 32),
R+       rep(0, 35), rep(1, 50))
R> y = c(rep(0, 84), rep(1, 85))
R> tab = table(x, y)
R> tab
```

```
   y
x    0  1
  0 52 35
  1 32 50
```

2.1. Mathematical Proof of AUC for Single Binary Predictor

As there are only two outcomes for X, we can expand the probability using the law of total probability:

$$\begin{aligned} P(X_1 > X_0) &= P(X_1 > X_0 | X_1 = 1) P(X_1 = 1) \\ &\quad + P(X_1 > X_0 | X_1 = 0) P(X_1 = 0) \qquad (1) \\ &= P(X_1 > X_0 | X_1 = 1) P(X_1 = 1) \qquad (2) \end{aligned}$$

where the second term of equation (1) is equal to zero because $X_0 \in \{0, 1\}$.
Here we see that the second term of equation (2) is the sensitivity:

$$\begin{aligned} P(X_1 = 1) &= P(X = 1|Y = 1) \\ &= \frac{TP}{TP + FN} \\ &= \text{sensitivity} \end{aligned}$$

Here we show the first term of equation (2) is the specificity:

$$\begin{aligned} P(X_1 > X_0|X_1 = 1) &= P(X_1 > X_0|X_1 = 1, X_0 = 1)P(X_0 = 1) \\ &+ P(X_1 > X_0|X_1 = 1, X_0 = 0)P(X_0 = 0) \\ &= P(X_1 > X_0|X_1 = 1, X_0 = 0)P(X_0 = 0) \\ &= P(X_0 = 0) \\ &= P(X = 0|Y = 0) \\ &= \frac{TN}{TN + FP} \\ &= \text{specificity} \end{aligned}$$

Therefore, we combine these two to show that equation (2) reduces to:

$$P(X_1 > X_0) = \text{specificity} * \text{sensitivity}$$

Therefore, the true AUC should be equal to: \begin{CodeChunk}
\begin{CodeInput} R> sens = tab[2,2] / sum(tab[,2]) R> spec = tab[1,1] / sum(tab[,1]) R> true_auc = sens * spec R> print(true_auc) \end{CodeInput}

```
[1] 0.3641457
```

\end{CodeChunk}

Reverse Labeling

\begin{CodeChunk}
\begin{CodeInput} R> flip_auc = (1 - sens) * (1 - spec) R> print(flip_auc) \end{CodeInput}

```
[1] 0.1568627
```

\end{CodeChunk}
\begin{CodeChunk}
\begin{CodeInput} R> fpr = 1-spec R> area_of_tri = 1/2 * sens * fpr R> area_of_quad = sens * spec + 1/2 * spec * (1-sens) R> auc = area_of_tri + area_of_quad \end{CodeInput}
\end{CodeChunk}
We can also show that if we use a simple sampling method, we can estimate this true AUC. Here, the function `est_auc` samples 10^{6} random samples from X_1 and X_0, then calculates $\hat{P}(X_1 > X_0)$:

\begin{CodeChunk}
\begin{CodeInput} R> est_auc = function(x, y) { R+ x1 = x[y == 1] R+ x0 = x[y == 0] R+ n = 1000000 R+ c1 = sample(x1, size = n, replace = TRUE) R+ c0 = sample(x0, size = n, replace = TRUE) R+ true_auc = mean(c1 > c0) R+ # calc_auc = true_auc + 1/2 * mean(c1 == c0) R+ # c(true_auc, calc_auc) R+ return(true_auc) R+ } R> sample_est_auc = est_auc(x, y) R> sample_est_auc \end{CodeInput}

```
[1] 0.364663
```

\end{CodeChunk}

3. Current Implementations

3.1. R

caTools Package

The **caTools** package is one of the most popular packages in R, and has analysis for doing area under the curve:

```
R> library(caTools)
R> colAUC(x, y)

             [,1]
0 vs. 1 0.6036415
```

ROCR Package

The **ROCR** package is one of the most popular packages for doing ROC analysis (Sing, Sander, Beerenwinkel, and Lengauer 2005). Using `prediction` and `performance` functions, we see that the estimated AUC is much higher than the true AUC:
\begin{CodeChunk}

```
R> library(ROCR)

Loading required package: gplots


Attaching package: 'gplots'

The following object is masked from 'package:stats':

    lowess
```

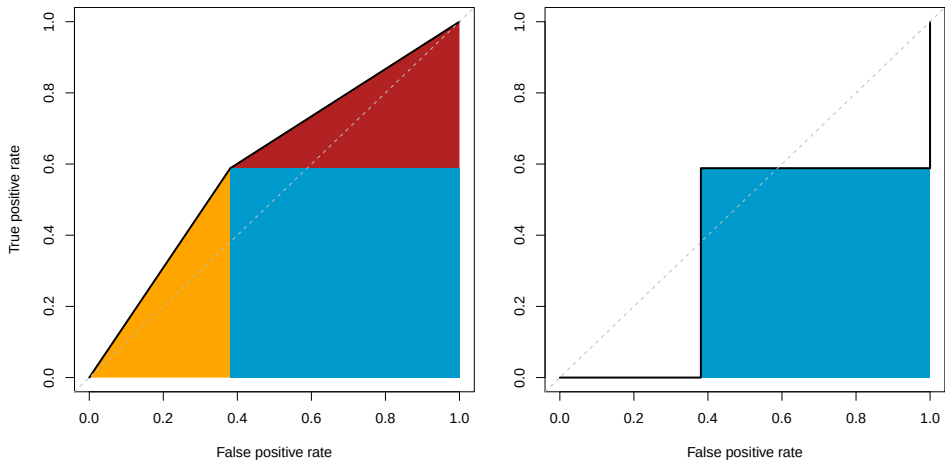
\begin{CodeInput} R> pred = prediction(x, y) R> auc_est = performance(pred, “auc”) R> auc_est@y.values[[1]] \end{CodeInput}

```
[1] 0.6036415
```

\end{CodeChunk}

Looking at the plot for the ROC curve in ROCR, we can see why this may be:

```
R> par(mfrow = c(1, 2))
R> perf = performance(pred, "tpr", "fpr")
R> plot(perf)
R> abline(a = 0, b = 1)
R> plot(perf, type = "s")
R> abline(a = 0, b = 1)
```



Looking geometrically at the plot, we can see how \begin{CodeChunk}

\begin{CodeInput} R> fpr = 1 - spec R> area_of_left_tri = 1/2 * sens * fpr R> area_of_top_tri = 1/2 * spec * (1 - sens) R> false_auc = area_of_left_tri + true_auc + area_of_top_tri R> false_auc \end{CodeInput}

```
[1] 0.6036415
```

\end{CodeChunk}

pROC Package

The **pROC** package is one of the most popular packages for doing ROC analysis (Robin, Turck, Hainard, Tiberti, Lisacek, Sanchez, and Müller 2011). Using `prediction` and `performance` functions, we see that the estimated AUC is much higher than the true AUC:

\begin{CodeChunk}

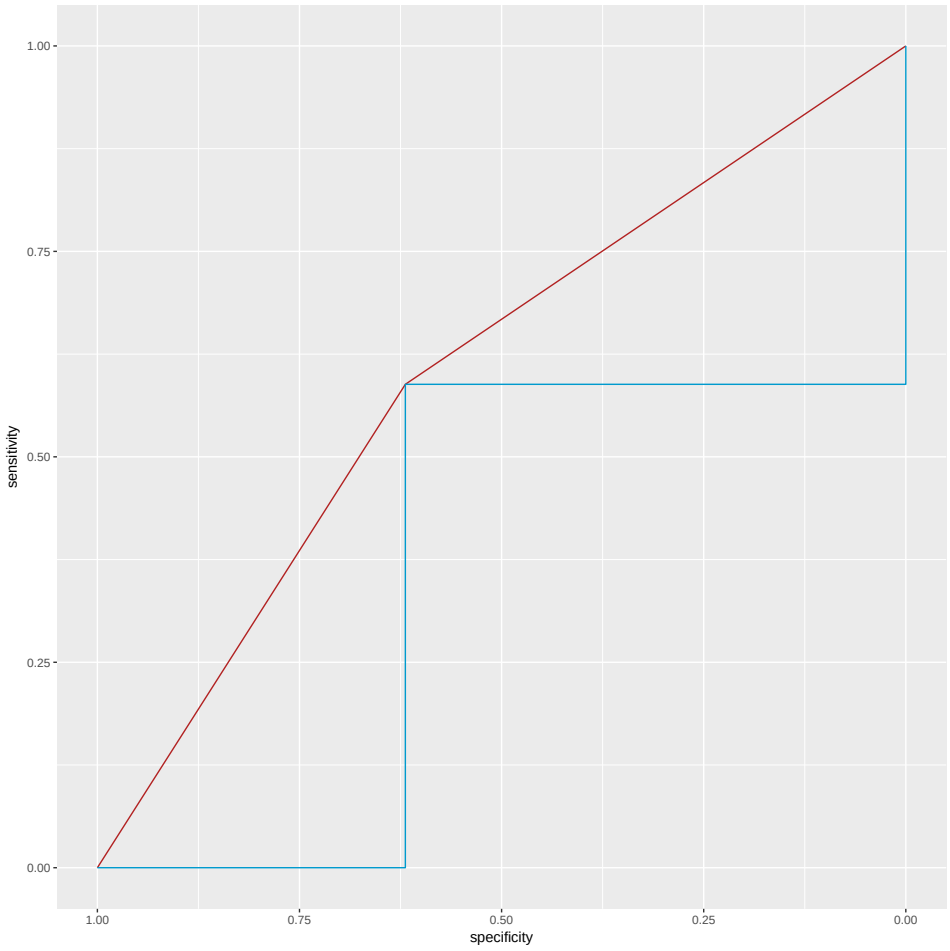
```
R> library(pROC)
```

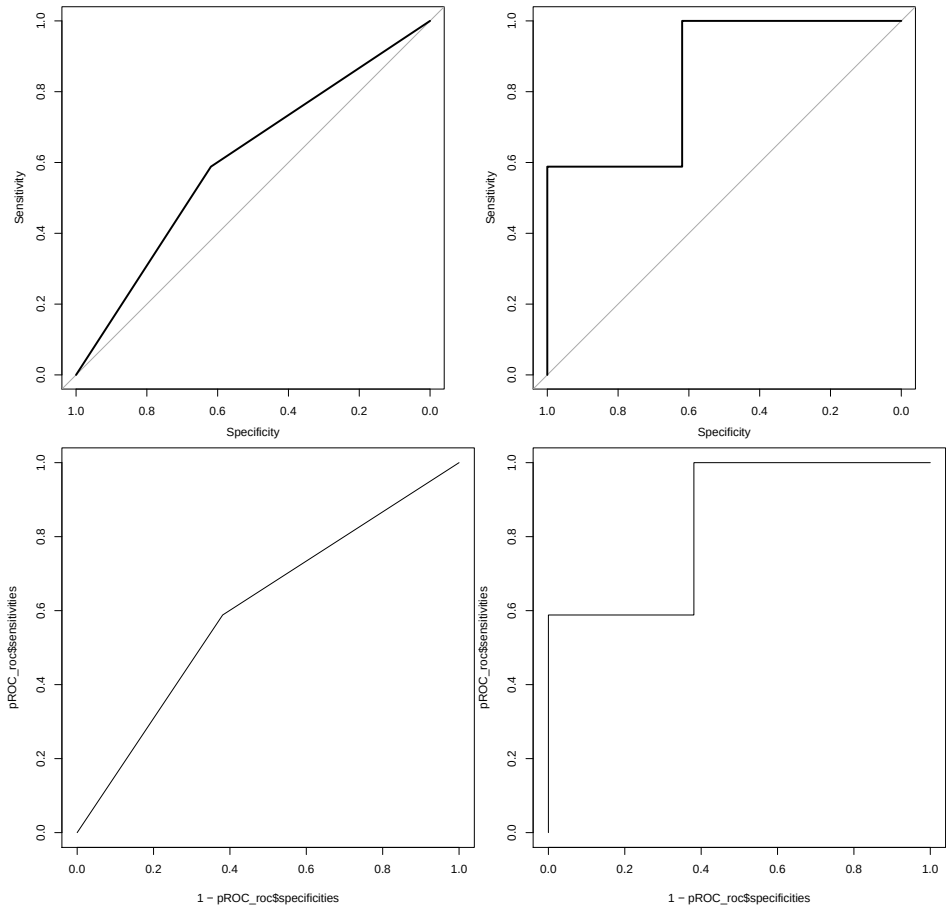
```
Type 'citation("pROC")' for a citation.


Attaching package: 'pROC'


The following objects are masked from 'package:stats':

    cov, smooth, var
```

\begin{CodeInput} R> pROC_roc = pROC::roc(predictor = x, response = y) R> pROC_roc[[“auc”]] \end{CodeInput}

```
Area under the curve: 0.6036
```

\end{CodeChunk}





fbroc Package

The **fbroc** package is one of the most popular packages for doing ROC analysis (Peter 2016). Using `prediction` and `performance` functions, we see that the estimated AUC is much higher than the true AUC:

\begin{CodeChunk}

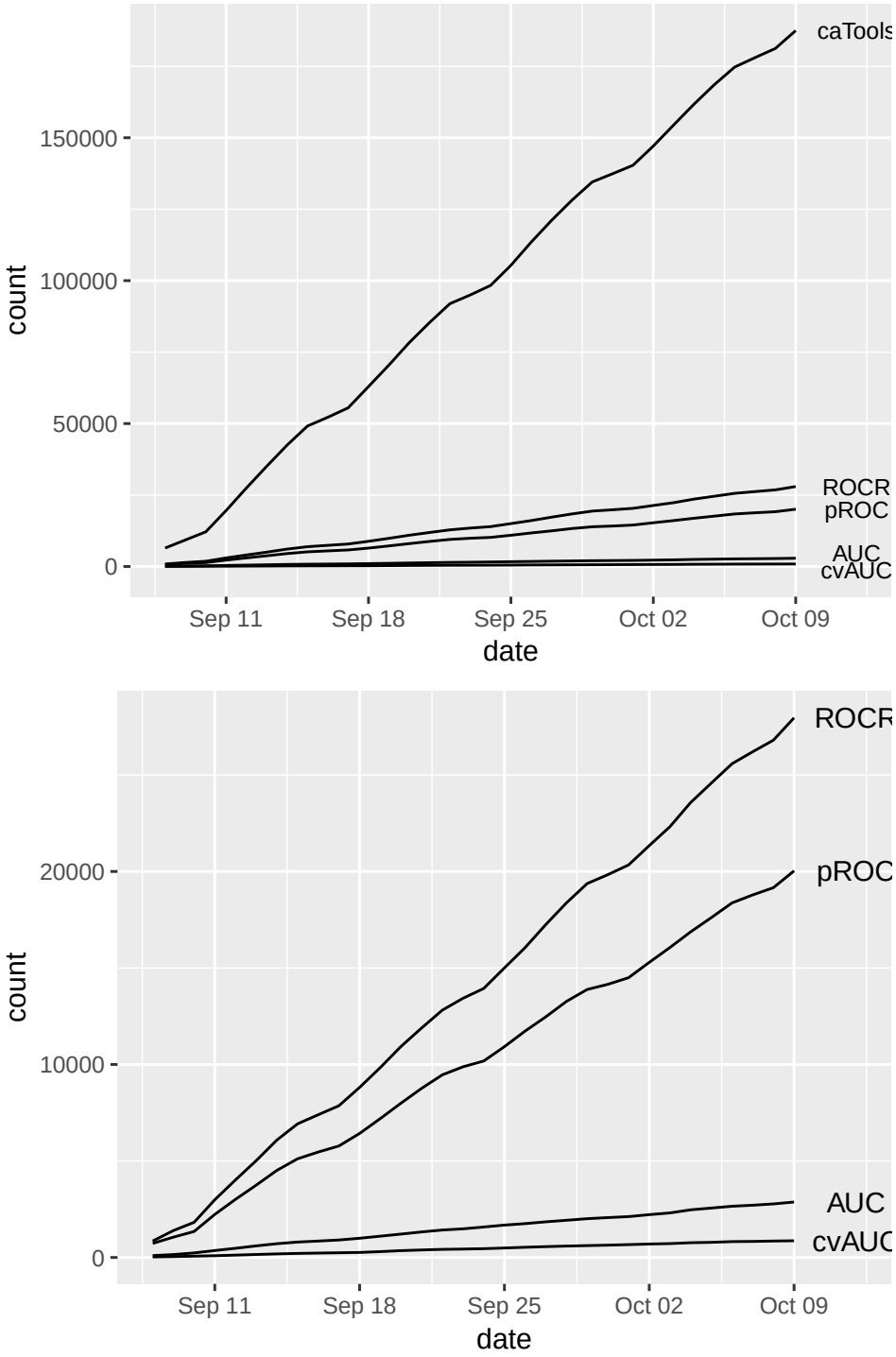
\begin{CodeInput} R> library(fbroc) R> fb_roc_default = boot.roc(x, as.logical(y), n.boot = 1000, tie.strategy = 2) R> auc_def = perf(fb_roc_default, “auc”) R> auc_def[[“Observed.Performance”]] \end{CodeInput}

```
[1] 0.6036415
```

\begin{CodeInput} R> fb_roc_alternative = boot.roc(x, as.logical(y), n.boot = 1000, tie.strategy = 1) R> auc_alt = perf(fb_roc_alternative, “auc”) R> auc_alt[[“Observed.Performance”]] \end{CodeInput}

```
[1] 0.6036415
```

\end{CodeChunk}



3.2. Stata

```
R> roctab x y
```

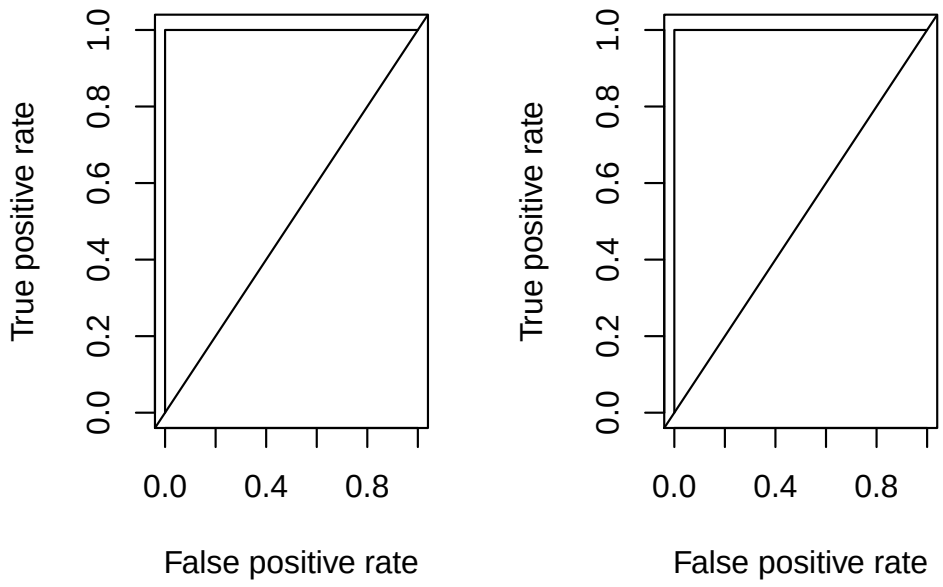
```
 . roctab x y
```

```
                ROC                    -Asymptotic Normal--
  Obs         Area      Std. Err.      [95% Conf. Interval]
-------------------------------------------------------------
  169       0.6037        0.0379        0.52952      0.67793
```

3.3. Fawcett Example

\begin{CodeChunk}

```
R> faw = data.frame(y = c(rep(TRUE, 6), rep(FALSE, 4)),
R+                  x = c(0.99999, 0.99999, 0.99993,
R+                        0.99986, 0.99964, 0.99955,
R+                        0.68139, 0.50961, 0.48880, 0.44951))
R> faw$hyp = faw$x > 0.5
R> pred = prediction(predictions = faw$x, labels = faw$y)
R> par(mfrow = c(1, 2))
R> perf = performance(pred, "tpr", "fpr")
R> plot(perf)
R> abline(a = 0, b = 1)
R> plot(perf, type = "s")
R> abline(a = 0, b = 1)
```



\begin{CodeInput} R> auc_est = performance(pred, “auc”) R> auc_est@y.values[[1]] \end{CodeInput}

```
[1] 1
```

\begin{CodeInput} R> est_auc(x = faw$x, y = faw$y) \end{CodeInput}

```
[1] 1
```

\end{CodeChunk}

\begin{CodeChunk}

\begin{CodeInput} R> library(dplyr) R> fawcett = function(df) { R+ L_sorted = df %>% R+ arrange(desc(x), y) R+ n_sample = nrow(L_sorted) R+ P = sum(L_sorted[["y"]]) R+ N = n_sample - P R+ FP = TP = 0 R+ R = NULL R+ f_prev = -Inf R+ i = 1 R+ for (i in seq(n_sample)) { R+ f_i = L_sorted[["x"]][i] R+ if (f_i != f_prev) { R+ fpr = FP/N R+ tpr = TP / P R+ R = rbind(R, c(fpr = fpr, tpr = tpr)) R+ f_prev = f_i R+ } R+ if (L_sorted$y[i]) R + TP = TP + 1 R+ R + if (!L_sorted$y[i]) { R+ FP = FP + 1 R+ } R+ } R+ fpr = FP/N R+ tpr = TP / P R+ R = rbind(R, c(fpr = fpr, tpr = tpr)) R+ return(R) R+ } R> fawcett_roc = fawcett(df) R> seq_range = function(x, ...) { R+ rx = range(x) R+ seq(rx[1], rx[2], ...) R+ } R> # seq_range(faw$x,) \end{CodeInput} \end{CodeChunk}

References

Fawcett T (2006). "An introduction to ROC analysis." *Pattern recognition letters*, **27**(8), 861–874.

Peter E (2016). *fbroc: Fast Algorithms to Bootstrap Receiver Operating Characteristics Curves*. R package version 0.4.0, URL https://CRAN.R-project.org/package=fbroc.

Robin X, Turck N, Hainard A, Tiberti N, Lisacek F, Sanchez JC, Müller M (2011). "pROC: an open-source package for R and S+ to analyze and compare ROC curves." *BMC Bioinformatics*, **12**, 77.

Sing T, Sander O, Beerenwinkel N, Lengauer T (2005). "ROCR: visualizing classifier performance in R." *Bioinformatics*, **21**(20), 7881. URL http://rocr.bioinf.mpi-sb.mpg.de.

Affiliation:

John Muschelli
Department of Biostatistics, Johns Hopkins Bloomberg School of Public Health
615 N Wolfe St Baltimore, MD 21205
E-mail: jmuschel@jhsph.edu
URL: http://johnmuschelli.

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd