

Revolutions

Milestones in AI, Machine Learning, Data Science, and visualization with R and Python since 2008

March 15, 2017

AUC Meets the Wilcoxon-Mann-Whitney U-Statistic

by Bob Horton, Senior Data Scientist, Microsoft

The area under an ROC curve (AUC) is commonly used in machine learning to summarize the performance of a predictive model with a single value. But you might be surprised to learn that the AUC is directly connected to the Mann-Whitney U-Statistic, which is commonly used in a robust, non-parametric alternative to Student's t-test. Here I'll use 'literate analysis' to demonstrate this connection and illustrate how the two measures are related.

In previous posts on [ROC curves](#) and [AUC](#) I described some simple ways to visualize and calculate these objects. Here is the simple data we used earlier to illustrate AUC:

```
category <- c(1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0)
prediction <- rev(seq_along(category))
prediction[9:10] <- mean(prediction[9:10])

library('pROC')
(official_auc <- auc(roc(category, prediction)))

## Area under the curve: 0.825
```

Here `category` is a vector of Boolean labels marking the true status of a sequence of cases. The other vector, `prediction`, represents a set of numeric scores as would normally be generated by some type of measurement or classifier algorithm. These scores could represent, for example, the expected probability of an object being a cat. But they don't need to be probabilities; any value indicating the relative strength of the classifier's confidence that the object is a cat can work, as long as the scores let us sort the cases into some order. Our fake scores are designed to put the cases in the order they start with, except that the scores of two cases have been replaced with their average; this gives us some instances where the scores are tied, which is a fairly reasonable condition we should be sure to handle. For this dataset the 'official' value for AUC is 0.825; when we try various other ways to calculate AUC, this is the number we want to see.

Computing AUC from the U Statistic

From [Wikipedia](#) we learn that

$$\{AUC\}_1 = \{U_1 \text{ over } n_1 n_2\}$$

where $\{U_1\}$ is the Mann-Whitney U statistic, also known as the Wilcoxon rank-sum test statistic, or some combination and/or permutation of those names. That seems like a strange claim, but it is easy enough to test:

```
auc_wmw <- function(labels, scores){
  labels <- as.logical(labels)
  pos <- scores[labels]
  neg <- scores[!labels]
  U <- as.numeric(wilcox.test(pos, neg)$statistic)
  U/(length(pos) * length(neg))
}

auc_wmw(category, prediction)

## Warning in wilcox.test.default(pos, neg): cannot compute exact p-value with
## ties

## [1] 0.825
```

The `wilcox.test` function warns us that we “cannot compute exact p-value with ties”, but for the current exercise let's just savor the

fact that it got the AUC exactly right.

To start to decipher the U statistic, let's calculate it ourselves instead of using the `wilcox.test` function as a black box. Going back to Wikipedia we learn that

$$U_1 = R_1 - \frac{n_1(n_1+1)}{2}$$

where R_1 is the sum of the ranks of the positive cases, and n_1 is the number of positive cases. You can calculate a related score (U_2) with the negative cases (of which there are n_2); these two scores are complementary in that they always add up to $\frac{n_1 n_2}{2}$, similar to the way that flipping positives and negatives when calculating an ROC curve gives you an AUC that is one minus the AUC of the unflipped curve. Anyway, now we have:

```
auc_wmw2 <- function(labels, scores){
  labels <- as.logical(labels)
  n1 <- sum(labels)
  n2 <- sum(!labels)
  R1 <- sum(rank(scores)[labels])
  U1 <- R1 - n1 * (n1 + 1)/2
  U1/(n1 * n2)
}
```

```
auc_wmw2(category, prediction)
```

```
## [1] 0.825
```

Base R's `rank` function assigns the lowest rank value (1, if there are no ties) to the lowest score, and by default it averages the ranks for tied scores, which is exactly what we need. Now I'll try to convince you that it makes sense that $U_1/(n_1 * n_2)$ is equal to AUC.

Graphical interpretation of the U Statistic

First we plot the ranks of all the scores as a stack of horizontal bars, and color them by the labels.

```
# simplify the syntax for drawing rectangles
rectangle <- function(x, y, width, height, density=12, angle=-45, ...)
  polygon(c(x,x,x+width,x+width), c(y,y+height,y+height,y),
    density=density, angle=angle, ...)

U_illustration_part1 <- function(labels, scores){
  # put cases in order by score
  sort_order <- order(scores)
  labels <- labels[sort_order]
  scores <- scores[sort_order]

  # count the cases
  n <- length(labels)

  # find overall rank for each case by score
  ranks <- rank(scores)

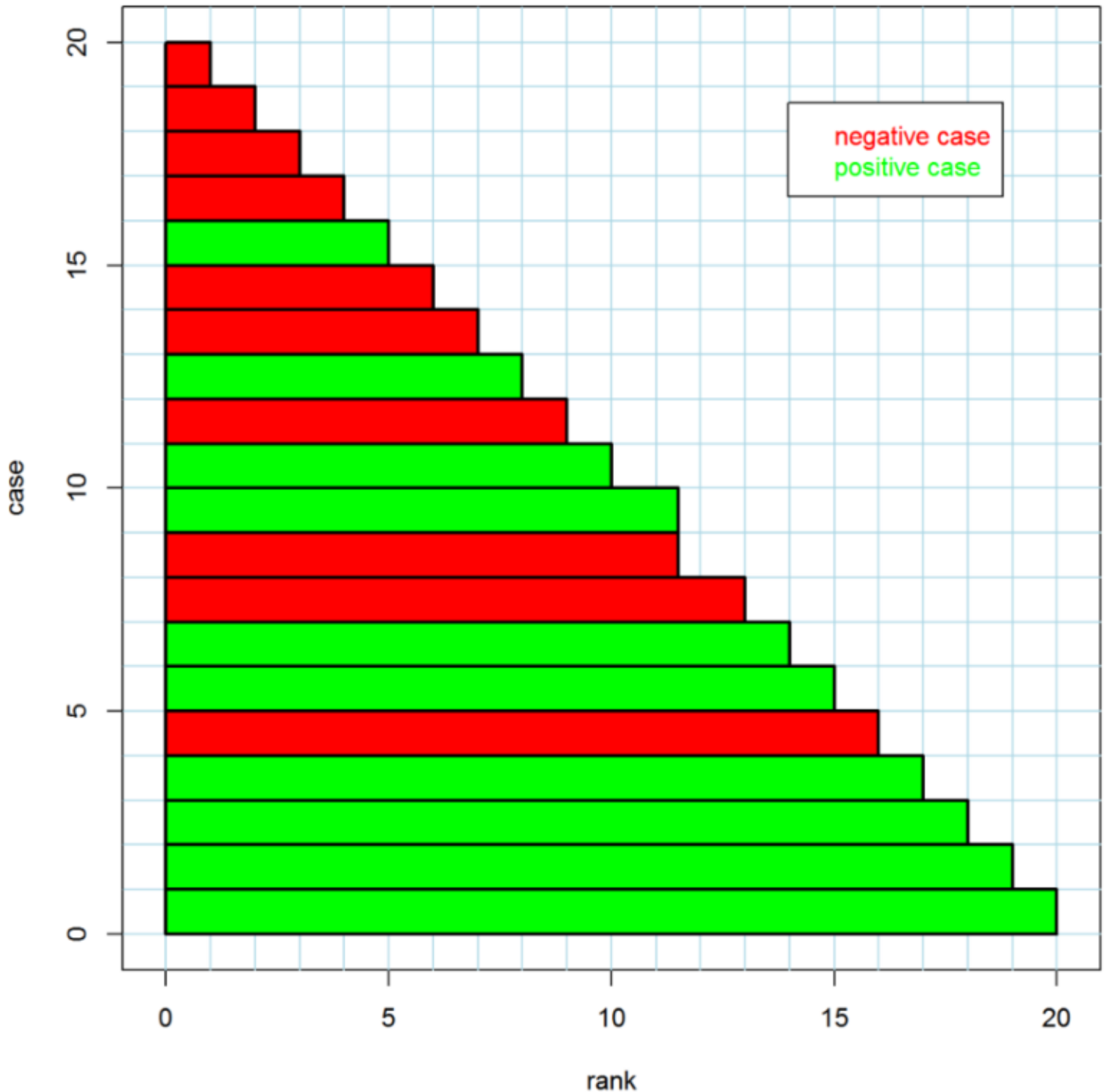
  # start with an empty plot
  plot(c(0, n), c(0, n), type='n',
    xlab="rank", ylab="case", asp=1)

  # draw a grid in the background
  abline(h=0:n, col="lightblue")
  abline(v=0:n, col="lightblue")
```

```
# plot bars representing ranks of all cases
mapply(rectangle, x=0, y=(n - 1):0, # starting from the top
        width=ranks, height=1,
        density=NA, border="black", lwd=2, col=c("red", "green")[1 + labels])

legend("topright", legend=c("negative case", "positive case"),
      text.col=c("red", "green"), bty='o', box.lwd=1, inset=0.1)
}

U_illustration_part1(labels=category, scores=prediction)
```



Now consider only the green bars, representing the ranks of the positive cases. We'll stack them on top of one another, and slide them horizontally as needed to get a nice even staircase on the right edge:

```
U_illustration_part2 <- function(labels, scores){
  # sort the cases
  sort_order <- order(scores)
  labels <- labels[sort_order]
  scores <- scores[sort_order]
```

```

# count positive and negative cases
n1 <- sum(labels) # number of positive cases
n2 <- sum(!labels) # number of negative cases

# find the overall ranks for the positive cases
ranks <- rank(scores)
pos_ranks <- ranks[as.logical(labels)]

# how far to slide each bar to make stairsteps on the right hand edge
x_offset <- n2 + (1:n1) - pos_ranks

# start with an empty plot
plot(c(0, n2 + n1), c(0, n1), type='n', asp=1,
      xlab="n2 + n1 divisions", ylab="n1 divisions")

# plot bars for ranks of positive cases
mapply(rectangle, x=x_offset, y=(n1 - 1):0,
        width=pos_ranks, height=1,
        density=NA, border="darkgreen", lwd=2, col="green")

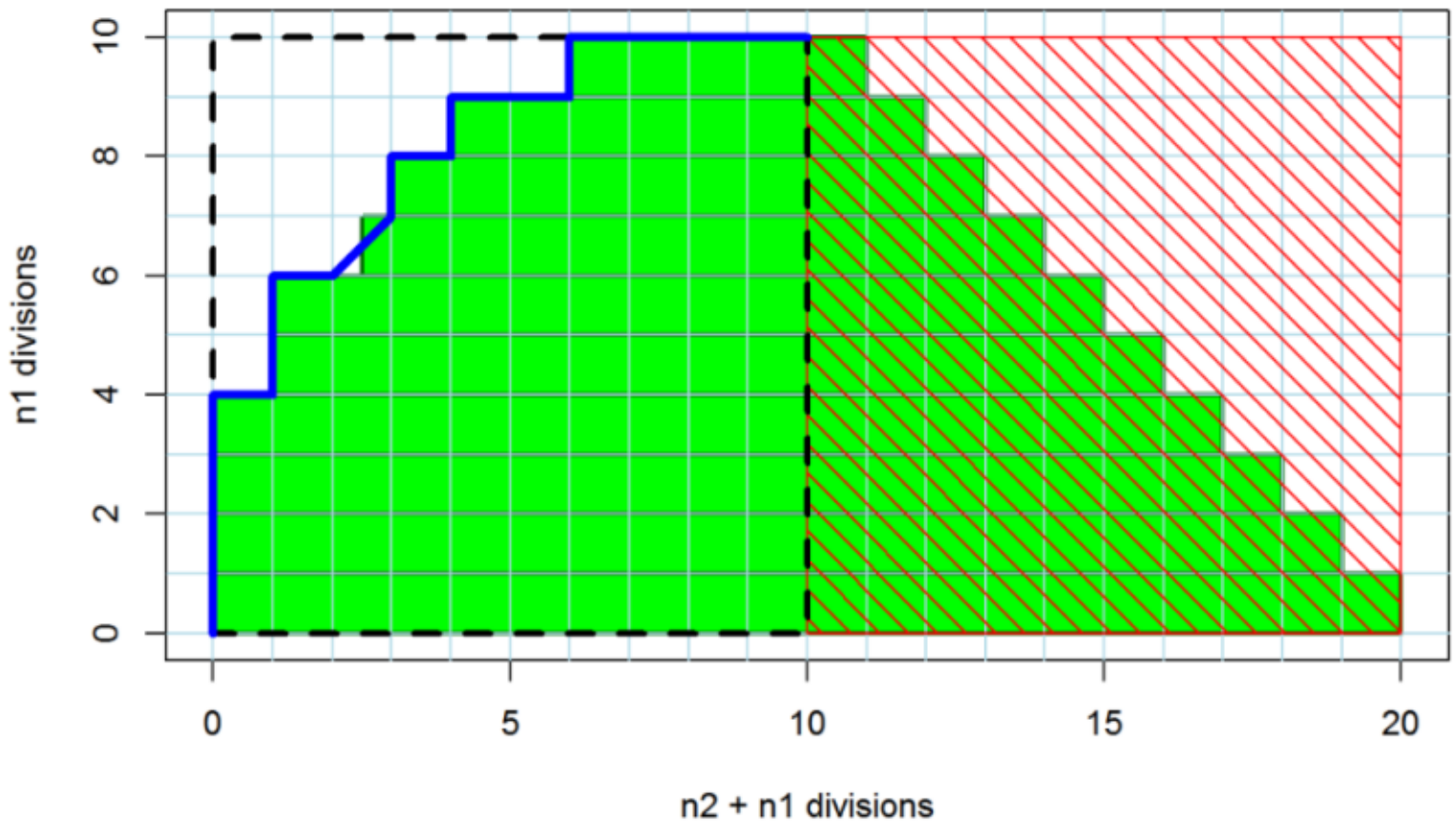
# draw the grid
abline(h=0:n1, col="lightblue")
abline(v=0:(n1 + n2), col="lightblue")

# mark the area we remove, and the area we keep
rectangle(n2, 0, n1, n1, density=10, col="red", lwd=1)
rectangle(0, 0, n2, n1, density=0, col="black", lty=2, lwd=3)

# draw a scaled version of the "official" ROC curve on top
roc_obj <- roc(labels, scores)
roc_df <- with(roc_obj, data.frame(FPR=rev(1 - specificities),
  TPR=rev(sensitivities)))
with(roc_df, lines(n2*FPR, n1*TPR, type='l', lwd=4, col="blue"))
}

U_illustration_part2(labels=category, scores=prediction)

```



The total area of all the green bars equals the sum of the ranks of the positive cases ($\sum(R_1)$ in the equation). The red hatched box on the right hand side represents the part we'll subtract. Note that the total area of green inside the red hatched box is $\frac{n_1(n_1+1)}{2}$, which you may recognize as the sum of the integers $\{1\}$ through $\{n_1\}$. The dashed rectangle on the left side shows the part we keep; the green area inside this dashed rectangle represents the value of U_1 .

Here n_1 , the number of steps on the y axis, is obviously just the number of green bars. It may take a bit more contemplation to see that n_2 , the number of steps on the x axis inside the dashed rectangle, reflects the number of red bars we removed (the negative cases). This plot is basically a transliteration of the formula for U_1 into a figure.

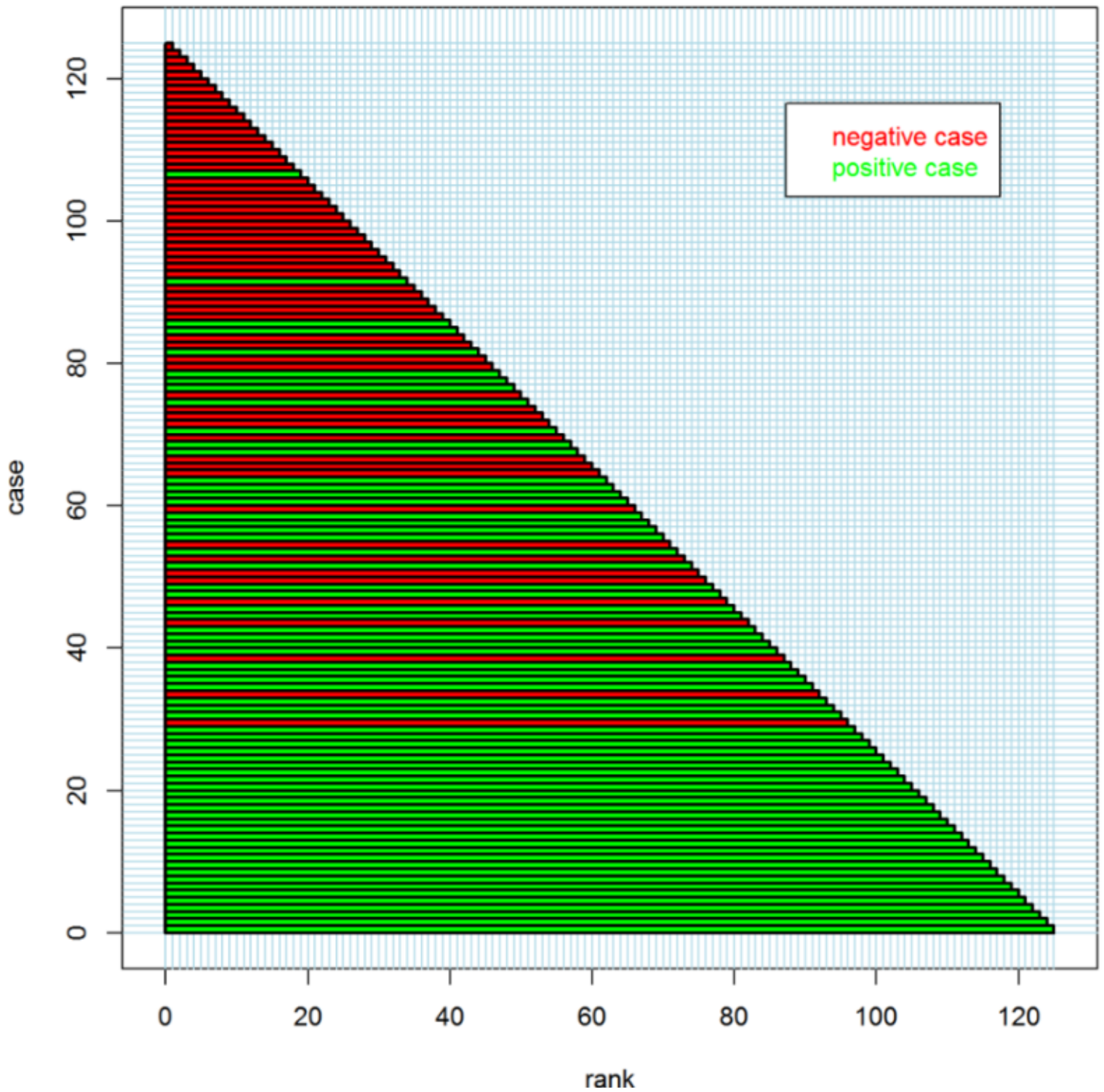
The whole grid of the dashed rectangle thus has an area of $n_1 * n_2$ steps squared, and U_1 is the area under the curve (in "square steps"). $U_1 / (n_1 * n_2)$ is the area under the curve as a fraction of the total area of the rectangle. In a traditional ROC curve the number of steps along each axis is normalized to 1, so that AUC is a fraction of a 1 by 1 area; once we normalize U_1 to the area of the (n_1) by (n_2) rectangle, it equals AUC.

The blue line is the 'official' ROC curve scaled by n_2 on the x (FPR) axis and by n_1 on the y (TPR) axis, and it matches the left edge of our stacked bars except where scores are tied. The half-unit area (where two scores are tied) is split vertically rather than diagonally, but it is still a half unit, so you can see that the area of stacked green bars within the dashed rectangle corresponds exactly to the area under the ROC curve.

Visualizing the U Statistic on a larger dataset

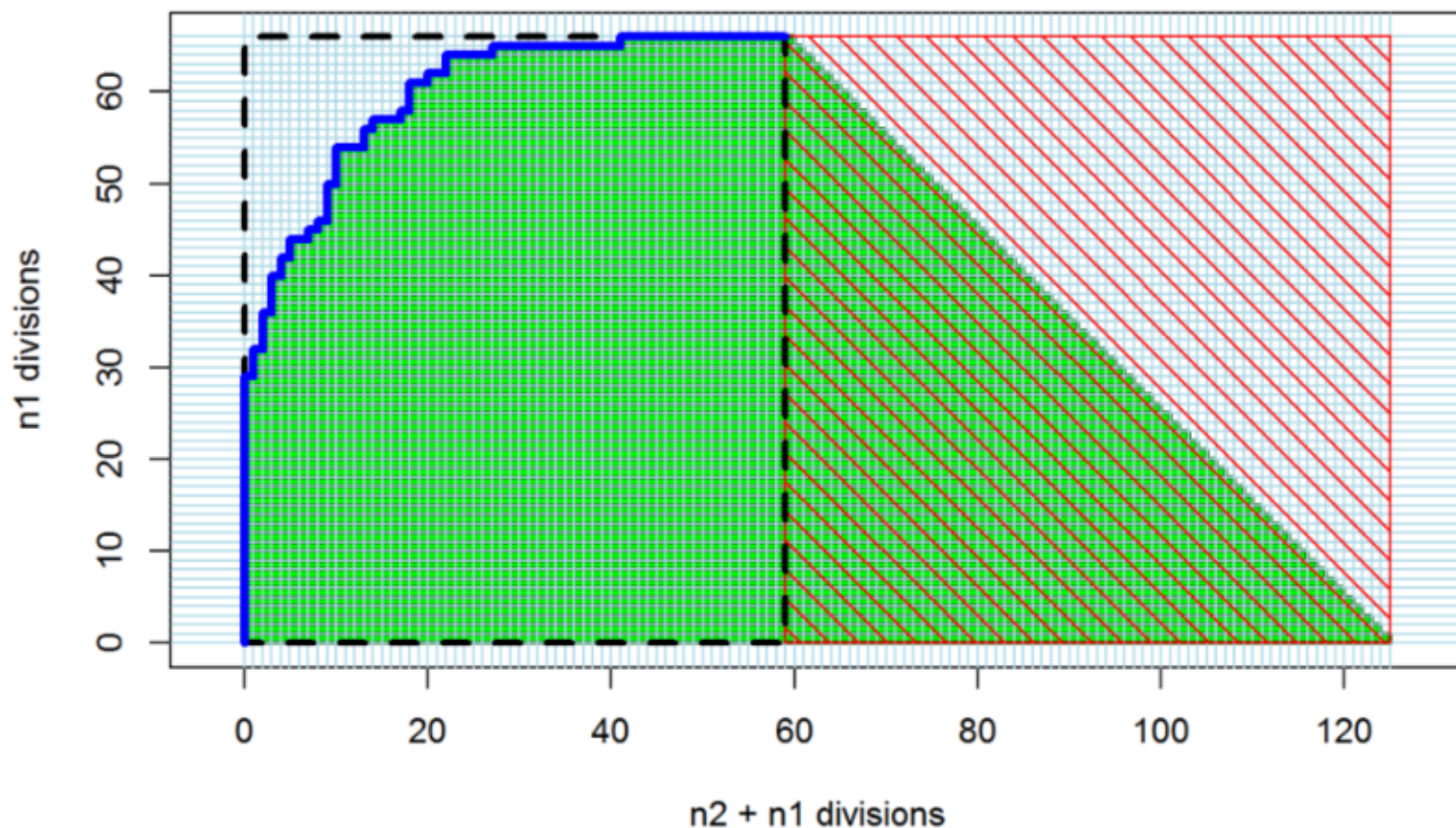
Let's generate similar figures using the example from the [earlier post](#), where the `test_set` dataframe has a Boolean column called `bad_widget` that labels cases, and a vector called `glm_response_scores` containing scores from a logistic regression. We'll use the same plotting code as above.

```
| U_illustration_part1(test_set$bad_widget, glm_response_scores)
```



Now we remove the red bars, stack the green ones together, and slide them to make even stair-steps on the right side. The left edge of these stacked bars forms an ROC curve.

```
| U_illustration_part2(test_set$bad_widgit, glm_response_scores)
```

Again, the blue line is the ‘official’ ROC curve calculated with the `pROC` package, then scaled by `n1` and `n2` on the y and x axes, respectively. In this dataset there are no tied scores so the ROC curve has no diagonal segments, and it matches the jagged contour of the left edge of our stacked rank bars exactly.

Check results with tied scores

Although this graphical approach does not draw the diagonal segments of an ROC curve correctly, calculation based on the U-statistic does get the AUC right even in the presence of ties. To demonstrate this more dramatically, we’ll round off the original scores to one decimal place so that many of the scores round to the same value, producing groups of tied scores (see the ROC plots for the rounded values in the earlier [AUC](#) post).

```
| rounded_scores <- round(glm_response_scores, digits=1)
```

Now we can use our U statistic - based functions to compute the AUC on both the original and rounded versions:

```
| data.frame(
|   auc_pROC = auc(roc(test_set$bad_widget, glm_response_scores)),
|   auc_wmw = auc_wmw(test_set$bad_widget == "TRUE", glm_response_scores),
|   auc_wmw2 = auc_wmw2(test_set$bad_widget == "TRUE", glm_response_scores),
|   auc_pROC_ties = auc(roc(test_set$bad_widget, rounded_scores)),
|   auc_wmw_ties = auc_wmw(test_set$bad_widget, rounded_scores),
|   auc_wmw2_ties = auc_wmw2(test_set$bad_widget, rounded_scores)
| )
| ##   auc_pROC  auc_wmw  auc_wmw2  auc_pROC_ties  auc_wmw_ties  auc_wmw2_ties
| ## 1 0.903698 0.903698 0.903698    0.8972779    0.8972779    0.8972779
```


Note that the “official” AUC is slightly smaller when the scores are rounded, and that both versions of our Wilcoxon-Mann-Whitney U-Statistic based functions get the AUC exactly right, with or without ties.

This post has focused on the value and interpretation of the **U statistic** itself, but we have not discussed the **U test**, which uses this

statistic (along with sample sizes) to evaluate statistical significance. According to [Wikipedia](#), the U test evaluates the null hypothesis that it is “equally likely that a randomly selected value from one sample will be less than or greater than a randomly selected value from a second sample.” As [illustrated earlier](#), AUC reflects a similar probability, that a randomly chosen positive case will receive a higher score from your model than a randomly chosen negative case. Although exploring how this test calculates p-values is beyond the scope of this post, it should not be surprising that metrics embodying these sorts of probabilities can form the basis of such tests.

Posted by [Guest Blogger](#) at 09:47 in [R](#), [statistics](#) | [Permalink](#)

Comments

 You can follow this conversation by subscribing to the [comment feed](#) for this post.

The comments to this entry are closed.