**Chapter 14: Online Appendices**

# Online Appendix 14.A: Proof of equivalence theorem for wAFROC

We prove the equivalence between the area under wAFROC plot and the definition of the wAFROC FOM. With reference to book Figure 14.1, the operating point labeled  has coordinates  given by book Equations 14.7 and 14.8, respectively, reproduced here for convenience:

 .

The area of the leftmost shaded trapezoid in Fig. 1 is:

 .

Then

 .

Using the probabilistic relation:

 .

we can expand the 1st term inside the square bracket:

 .

The last two terms are equal, therefore:

 .

The final steps of the proof require that the z-samples be converted to integer ratings, which can be done without loss of ordering information if the number of bins is sufficiently large. Let  denote the integer rating of mark , which implies that marks with z-samples satisfying ,  , are rated  (binning rule). The zero rating corresponds to unmarked regions. The dummy thresholds  and  are defined as  and , respectively. From Eqn. (15) it follows that

 .

Because of the binning rule,  can be replaced by,  can be replaced by  and can be replaced by . Then Eqn. (16) can then be re-written as:



Summing over all values of , one gets for the total area under the empirical wAFROC plot:

 .

Explanation: Eqn. (18) follows from the property of the indicator function, which constrains *i* in the indicator functions inside the square bracket in Eqn. (17) to , where the functions are unity and otherwise they are zero. Using the definition of the Wilcoxon kernel function, Eqn. (3), it follows that:

 .



## Online Appendix 14.A.1: Contribution of *i* = 0 term:

We prove a theorem that the contribution of the *i* = 0 term in Eqn. TBA is identical to the area under the extension of the wAFROC from the uppermost non-trivial point to (1,1). According to Eqn. TBA,

 .

Rearranging the summations:

 .

The following relation is easily shown:

 .

This is because the indicator function and the summation symbol counts the numbers of unmarked (zero rated) non-diseased cases and the division by  yields the corresponding contribution to *FPF*, which is the complement of the largest FPF value, , obtained by cumulating all ratings 1 and above. Similarly,

 .

Also,

 .

Using these expressions, Eqn. TBA reduces to:

 .

This expression can be simplified to:

 .

This is seen to be the area of a rectangle of base and height  plus the area of a triangle of baseand height , in other words it is the area under the extension of the wAFROC from the uppermost non-trivial point to (1,1).



# Online Appendix 14.B: Understanding the AFROC and wAFROC

The code in file mainExamples.R simulates a dataset, line 4, consisting of  non-diseased cases and  diseased cases. The numbers of lesions per diseased case Lk2 are determined at line 5. The simulation parameters defined at line 6 are mu = 2, lambda = 1, nu = 1 and zeta1 = ‑1. The first two diseased cases have one lesion each, and each of the remaining two have two lesions (after sourcing the code, highlight Lk2 and click Run. It yields the plots shown in book Figure 14.3 and 14.5 (without the labeling) and the values of the AFROC and wAFROC FOMs.

### Online Appendix 14.B.1: Code Listing

rm(list = ls()) # mainExamples.R

require(RJAfroc)

seed <- 1;set.seed(seed)

K1 <- 4;K2 <- 4

maxLL <- 2;Lk2 <- ceiling(runif(K2) \* maxLL)

mu <- 2;lambda <- 1;nu <- 1 ;zeta1 <- -1

frocData <- FROCSimulator(mu = mu, lambda = lambda, nu = nu,

K1 = K1, K2 = K2, Lk2 = Lk2, zeta1 = zeta1)

FP <- apply(frocData$NL, 3, max);FP <- FP[1:K1]

frocData$lesionWeight[3, ] <- c(0.6, 0.4)

frocData$lesionWeight[4, ] <- c(0.4, 0.6)

plotAfroc <- EmpiricalOpCharac(frocData,trts = 1, rdrs = 1, opChType = "AFROC", lgdPos = "NULL")

print(plotAfroc$AFROCPlot)

plotwAfroc <- EmpiricalOpCharac(frocData,trts = 1, rdrs = 1, opChType = "wAFROC", lgdPos = "NULL")

print(plotwAfroc$wAFROCPlot)

cat("AFROC AUC = ", FigureOfMerit(frocData, FOM = "AFROC"),"\n")

cat("wAFROC AUC = ", FigureOfMerit(frocData, FOM = "wAFROC"),"\n")

Line 11 - 12 assigns the non-default values of the lesion weights (the default value for two lesions per case would be 0.5 to each lesion). The remaining lines plot the relevant curves. The reader should try different lesion weights, and perhaps increase the number of cases and compare the curves and AUC values.

### Online Appendix 14.B.2: Code output

> source(...)

AFROC AUC = 0.7708333

wAFROC AUC = 0.7875

Line 11 - 12 assigns the non-default values of the lesion weights (the default value for two lesions per case would be 0.5 for each lesion). The remaining lines plot the relevant operating characteristics.

# Online Appendix 14.C: Summary of FROC FOMs

Summarized in Table 1 are formulae for all AFROC-based empirical FOMs that have appeared in the literature. Note that the AFROC1 and wAFROC1 FOMs include comparisons between LL and NL z-samples on diseased cases, which are not allowed in the AFROC and wAFROC FOMs. Excepting for the ROC FOM, all AFROC-based FOMs are contained in the interval [0,1]. The ROC FOM is contained in the interval [0.5,1].

Table : Definitions of all currently implemented AFROC-based FOMs in JAFROC and RJAfroc software. Note that we do not use the prefix J with these FOMs, e.g., JAFROC 🡪 AFROC, wJAFROC 🡪 wAFROC, etc. A FOM name ending in 1 implies that *all* highest-rated NLs are used in computing the relevant statistic, including those on diseased cases.

|  |  |  |
| --- | --- | --- |
| **FOM name** | **Definition** | **Comments** |
| AFROC |  |  |
| wAFROC |  | Recommended when one has both diseased and non-diseased cases |
| AFROC1 |  |  |
| wAFROC1 |  | Recommended only when one does not have any non-diseased cases, i.e., |
| Inferred  ROC |  |  |
| FROC |  | Not-recommended |

# Online Appendix 14.D: Numerical demonstrations of FOM vs. AUC equivalences

The purpose of this section is to demonstrate numerically that the trapezoidal area under an operating characteristic equals the appropriate FOM formula defined in Table 1.

Note the usage of the R package caTools, which contains the function trapz to calculate the trapezoidal area. If one gets a message like there is no package called ‘caTools’, it means the relevant package has not been loaded. Use the Packages – Install tabs in the lower right window of RStudio window to load the missing package, Figure 1.

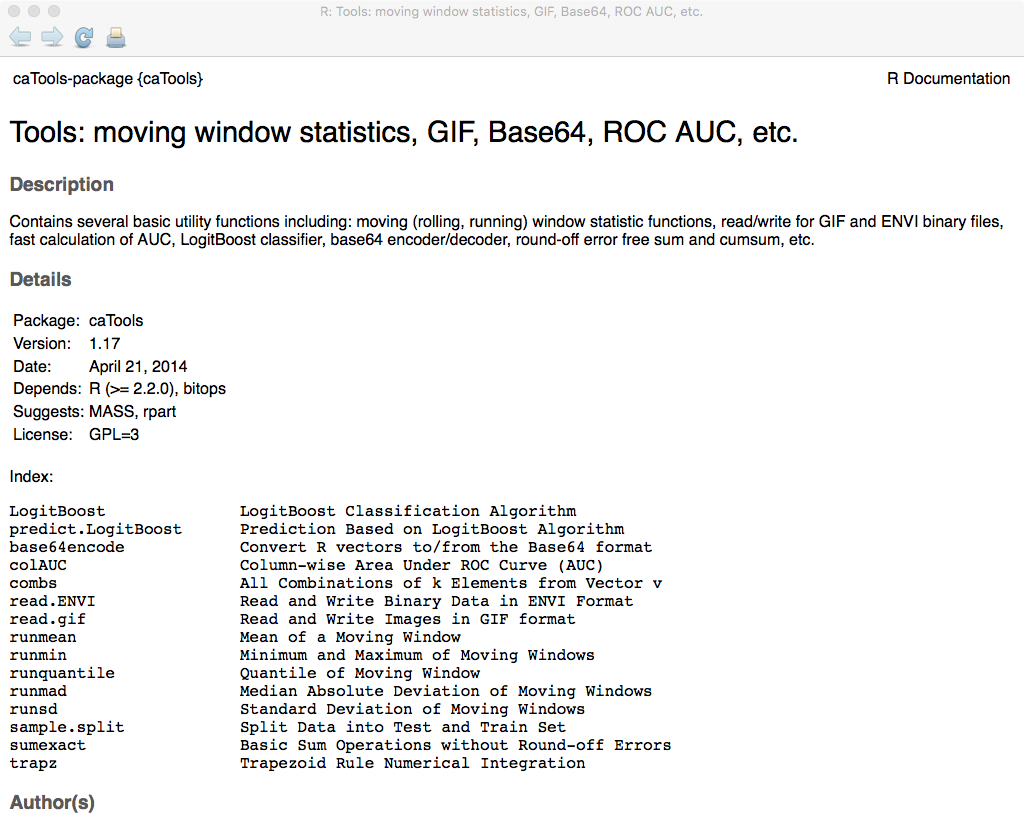


Figure : Help screen for caTools package. The function we are interested in is trapz.

The code in mainFOMvsAUC.R is shown below:

### Online Appendix 14.D.1: Code Listing

rm(list = ls()) # MainFOMvsAUC.R; demonstration of equivalences between FOMs and AUCs

library(RJafroc)

library(caTools)

seed <- 1;set.seed(seed)

Lmax <- 2

# following generates 1 to Lmax lesions per dis. case

K1 <- 7;K2 <- 8;Lk2 <- ceiling(runif(K2, 0, Lmax))

cat("K1 = ", K1, ", K2 = ", K2, "\n")

mu <- 1.5;lambda <- 0.8;nu <- 0.8 ;zeta1 <- -1

frocDataRaw <- SimulateFrocDataset(

mu = mu,

lambda = lambda,

nu = nu,

I = 1,

J = 1,

K1 = K1, K2 = K2, lesionNum = Lk2, zeta1 = zeta1)

# compare afrocAUC vs. afrocFOM

# compare afrocAUC vs. afrocFOM

afrocPlot <- PlotEmpiricalOperatingCharacteristics(

frocDataRaw,1,1,opChType = "AFROC")

afrocFOM <- signif(as.numeric(UtilFigureOfMerit(frocDataRaw, FOM = "AFROC")), digits = 8)

# trapz(x, y) function, below, returns the trapezoid integral defined by x and y,

# which can be used to calculate the trapezoidal area in our example. Use signif() function

# to round the results to 8 significant digits

afrocAUC <- signif(trapz(afrocPlot$Points$genAbscissa, afrocPlot$Points$genOrdinate), digits = 8)

print(afrocAUC == afrocFOM)

cat("afrocFOM, afrocAUC = ", afrocAUC, "\n")

# compare wafrocAUC vs. wafrocFOM

# compare wafrocAUC vs. wafrocFOM

wafrocPlot <- PlotEmpiricalOperatingCharacteristics(frocDataRaw,1,1,opChType = "wAFROC")

wafrocFOM <- signif(as.numeric(UtilFigureOfMerit(frocDataRaw, FOM = "wAFROC")), digits = 8)

wafrocAUC <- signif(trapz(wafrocPlot$Points$genAbscissa, wafrocPlot$Points$genOrdinate), digits = 8)

print(wafrocAUC == wafrocFOM) # shows TRUE if equality is satisfied

cat("wafrocFOM, wafrocAUC = ", wafrocAUC, "\n")

# compare afroc1AUC vs. afroc1FOM

# compare afroc1AUC vs. afroc1FOM

afroc1Plot <- PlotEmpiricalOperatingCharacteristics(frocDataRaw,1,1,opChType = "AFROC1")

afroc1FOM <- signif(as.numeric(UtilFigureOfMerit(frocDataRaw, FOM = "AFROC1")), digits = 8)

afroc1AUC <- signif(trapz(afroc1Plot$Points$genAbscissa, afroc1Plot$Points$genOrdinate), digits = 8)

print(afroc1AUC == afroc1FOM) # shows TRUE if equality is satisfied

cat("afroc1FOM, afrocAUC1 = ", afroc1AUC, "\n")

# compare wafroc1AUC vs. wafroc1FOM

# compare wafroc1AUC vs. wafroc1FOM

wafroc1Plot <- PlotEmpiricalOperatingCharacteristics(frocDataRaw,1,1,opChType = "wAFROC1")

wafroc1FOM <- signif(as.numeric(UtilFigureOfMerit(frocDataRaw, FOM = "wAFROC1")), digits = 8)

wafroc1AUC <- signif(trapz(wafroc1Plot$Points$genAbscissa, wafroc1Plot$Points$genOrdinate), digits = 8)

print(wafroc1AUC == wafroc1FOM) # shows TRUE if equality is satisfied

cat("wafroc1FOM, wafroc1AUC = ", wafroc1AUC, "\n")

# compare frocAUC vs. frocFOM

# compare frocAUC vs. frocFOM

frocPlot <- PlotEmpiricalOperatingCharacteristics(frocDataRaw,1,1,opChType = "FROC")

frocFOM <- signif(as.numeric(UtilFigureOfMerit(frocDataRaw, FOM = "FROC")), digits = 8)

frocAUC <- signif(trapz(frocPlot$Points$genAbscissa, frocPlot$Points$genOrdinate), digits = 8)

print(frocAUC == frocFOM) # shows TRUE if equality is satisfied

cat("frocFOM, frocAUC = ", frocAUC, "\n")

Line 5 – 17 simulates an FROC dataset consisting of 7 non-diseased and 8 diseased cases. The parameters of the simulation model are defined at line 10. Insert a break point at line 21 and Source the code, yielding the following output.

### Online Appendix 14.D.2: Debug Output 1

> debugSource(...)

K1 = 7 , K2 = 8

...

Highlight Lk2 and click Run, yielding:

### Online Appendix 14.D.3: Debug Output 1

Browse[2]> Lk2

[1] 1 1 2 2 1 2 2 2

The first two diseased cases have one lesion each; the next two have 2 lesions each, etc. Lines 19 – 29 test the equality of the AFROC-AUC and the AFROC FOM. Continue clicking Next until the code pointer advances to line 33. The result of executing line 28 is printed as TRUE, meaning the numerical trapezoidal integration, performed at line 27, using function trapz(), yields the same result as the FOM formula, coded in function UtilFigureOfMerit(), which was called at line 23. Line 29 prints the common value, 0.8021978, for this simulation. Bottom line, for this simulation the trapezoidal AUC equals that yielded by the FOM formula.

Likewise, lines 31 – 37 test the equality of the wAFROC-AUC and the wAFROC FOM; lines 39 – 45 test the equality of the AFROC1-AUC and the AFROC1 FOM; lines 47 – 53 test the equality of the wAFROC1-AUC and the wAFROC1 FOM; lines 55 – 61 test the equality of the FROC-AUC and the FROC FOM.

Remove any breakpoints and source the code, yielding.

### Online Appendix 14.D.4: Code Output 1

> source(...)

K1 = 7 , K2 = 8

[1] TRUE

afrocFOM, afrocAUC = 0.8021978

[1] TRUE

wafrocFOM, wafrocAUC = 0.7991071

[1] TRUE

afroc1FOM, afrocAUC1 = 0.774359

[1] TRUE

wafroc1FOM, wafroc1AUC = 0.7729167

[1] TRUE

frocFOM, frocAUC = 0.4

Each TRUE is the result of a logical comparison between a FOM and the corresponding AUC; the accuracy of the comparison is 8 significant digits. The common value is then printed out. For example, . The reader should experiment with the code to confirm that it always yields TRUE for the listed comparisons.