**Chapter 17: Predictions of the RSM, Online Appendices**

# Table of Contents

Online Appendix 17.A: The error function

Online Appendix 17.B: Derivation of expression for TPF

Online Appendix 17.C: Expression for pdf of diseased cases

Online Appendix 17.D: RSM-predicted ROC & pdf curves

Online Appendix 17.E: Is FROC good?

Online Appendix 17.F: Binormal parameters for RSM-generated ROC datasets

Online Appendix 17.G: Explanations of Swets et al observations

Online Appendix 17.H: On the continuity of the slope of the RSM-predicted ROC curve at the end-point

# Online Appendix 17.A: The error function in R

The error function is defined by1

 .

From the definition of the unit normal CDF function, it follows that:



The final result is:

 .



# Online Appendix 17.B: Derivation of expression for TPF

Since it is a little complicated, and Maple needed the author's help in the simplification, the code and the simplification are shown below (of course, one needs access to Maple to run this code).

## Online Appendix 17.A.1: Maple Code

phi := proc(t,mu) (1/sqrt(2\*Pi))\*exp(-(t-mu)^2/2); end:

PHI := proc(c,mu) local t; int(phi(t,mu),t=-infinity..c); end:

Poisson := proc(n, lambda) lambda^n \* exp(-lambda) / n!;end:

Bin := proc(l,N,nu) binomial(N,l) \* nu^l \* (1-nu)^(N-l);end:

TPF\_n\_l := proc(zeta,mu,n,l) 1 - PHI(zeta,0)^n\*PHI(zeta,mu)^l; end:

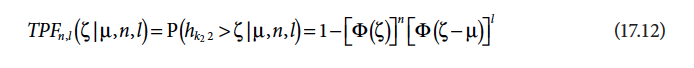
TPF\_n\_N := proc(zeta,mu,nu,n,N) sum(Bin(l,N,nu)\*TPF\_n\_l(zeta,mu,n,l),l=0..N); end:

TPF := proc(zeta,mu,lambda,nu,N) sum(Poisson(n,lambda)\*TPF\_n\_N(zeta,mu,nu,n,N),n=0..infinity);end:

TPF(zeta,mu,lambda,nu,N);

The first four lines define the probability density function corresponding to , the cumulative distribution function corresponding to , the Poisson distribution probability mass function for Poisson parameter , and the binomial distribution probability mass function corresponding to success probability  and trial size *N*, respectively[[1]](#footnote-1).

The sixth line implements book Equation 17.12, the contribution to *TPF* at threshold  of diseased cases with *n* latent NLs and *l* latent LLs:



The next line implements the binomial weighted summation over *l* (L is the total number of lesions per diseased case, assumed constant)

 .

The last line implements the Poisson weighted summation over :

 .

The final line displays the result:



This looks complicated, and probably someone with greater experience with Maple can write the appropriate code to simplify it[[2]](#footnote-2). I will do it the old fashioned way. The last term (in red font) is easily seen to be unity. So one is left with:



Factoring out  from the denominator of the large bracketed term, one gets:



It is seen that the term cancels out leaving:



Simplifying this leads to:

 .

And one last simplification (the error function is anti-symmetric in its argument):

 .

This is identical to Eqn. 9 in Ref. 2, reproduced here for convenience of comparison (with obvious replacements  corresponding to the new re-parameterized notation):

 .

Equation is identical to book equation 17.14.



# Online Appendix 17.C: Expression for pdf of diseased cases

For convenience, using intermediate variables *A* and *B*, defined as follows:

 .

 .

It can be shown that:

 .



# Online Appendix 17.D: RSM-predicted ROC & pdf curves

The following code, which generates pdf plots for both truth states, was used to generate book Figure 17.10 (a – d), which explain the empirical observations that the b-parameter of the binormal model are usually less than unity, i.e., the diseased case pdf is wider than the non-diseased case pdf.

## Online Appendix 17.D.1: Code listing

# mainRsmPlots.R

rm(list = ls())

library(RJafroc)

library(ggplot2)

K2 <- 700;Lmax <- 4;Lk2 <- floor(runif(K2, 1, Lmax + 1))

nLesPerCase <- unique(Lk2);lesionDist <- array(dim = c(length(nLesPerCase), 2))

for (i in nLesPerCase) lesionDist[i, ] <- c(i, sum(Lk2 == i)/K2)

muArr <- c(2,3);lambda <- 1;nuArr <- c(0.15,0.25); L <- 1 # to show wider pdfs of diseased cases

for (i in 1:length(muArr)) {

mu <- muArr[i]

for (j in 1:length(nuArr)) {

nu <- nuArr[j]

ret1 <- PlotRsmOperatingCharacteristics(

mu, lambda, nu,

type = "ALL", lesionDistribution = lesionDist, legendPosition = "none"

)

pdfPlots <- ret1$PDFPlot +

scale\_color\_manual(values = c("black","darkgrey"), guide = FALSE) +

theme(axis.title.y = element\_text(size = 25,face="bold"),

axis.title.x = element\_text(size = 30,face="bold"),

legend.title = element\_blank(),

legend.position = c(0.77,0.95),

legend.direction = "horizontal",

legend.text = element\_text(size = 20, face = "bold"),

legend.key.size = unit(2, "lines")) +

scale\_x\_continuous(expand = c(0, 0)) +

scale\_y\_continuous(expand = c(0, 0))

pdfPlots$layers[[1]]$aes\_params$size <- 2 # line

print(pdfPlots)

cat("mu =", mu, ", lambda =", lambda,", nu =", nu,"\n")

next

}

}

The option lesionDist, calculated at line 6-8, and supplied to RsmOperatingCharacteristics() at line 17, lesionDistribution = lesionDist, is the normalized histogram of the numbers of lesions per case, which specifies the fraction of diseased cases having one lesion, two lesions, three lesions, etc. To keep it simple, the code is restricted to one lesion per diseased case, so lesionDist = c(1,1). However, the reader is encouraged to try different integer values for . For example, with  = 4, lesionDist = [1:4,1:2] matrix, where each row contains the number of lesions (1:4) and the fraction of diseased cases with that many lesions. The reader might wonder why, when random sampling is not involved in the code, does one need to specify K2? A large value (700) at line 6 is specified so that the lesion distribution matrix is reasonably accurate. With a small value for K2 the normalized histogram will be "choppy" and introduce sampling variability, and with few enough cases, the desired distribution cannot be achieved, as illustrated in the code below.

> K2

[1] 700

> lesionDist

[,1] [,2]

[1,] 1 0.2557143

[2,] 2 0.2514286

[3,] 3 0.2728571

[4,] 4 0.2200000

> K2

[1] 70

> lesionDist

[,1] [,2]

[1,] 1 0.30000000

[2,] 2 0.08571429

[3,] 3 0.34285714

[4,] 4 0.27142857

> K2

[1] 7

Error in `[<-`(`\*tmp\*`, i, , value = c(i, sum(Lk2 == i)/K2)) :

subscript out of bounds

> nLesPerCase

[1] 1 3 4

Using the first example (700) the above output shows that 25.6% of diseased cases contain one lesion, 25.1% contain two lesions, 27.3% contain three lesions and 22% contain four lesions. These numbers add up to unity. A similar finding is obtained with 70 cases. But with 7 cases the program yields an error, because there are no cases with two lesions. In other words the histogram has zero height for 2 lesions. This is the reason the dummy variable K2 is chosen to be large.

Line 17 specifies type = "ALL" (meaning *all* types of plots, ROC, FROC, and AFROC are returned; see RJafroc documentation) and the result is saved to ret1. Line 19-31 displays the pdf plot.

To plot a curve one extracts the *correct* plot from ret1 and prints it. As an example, to display an AFROC plot, the plot object is ret1$AFROCPlot and one uses print(ret1$AFROCPlot) to display it (try it!). Incidentally, one is not "eternally wedded" to the documentation files. Type ret1 at the Console prompt and watch what happens when one types $, meaning one is looking for a list member in the ret1 object; a pop-up window appears with a number of choices allowing one to select pdfPlot or ROCPlot: this is RStudio is working in the background, trying to help.

Source the code to yield book Figure 17.10 (a – d).

## Online Appendix 17.D.2: Code listing

This relates to book section 17.5, i.e., RSM-predicted ROC and pdf curves. The code in mainRsmPdfRoc.R displays pdfs and ROC plots for the values of  specified in muArr = c(0.001, 1, 2, 3, 4 and 5) at line 6. The remaining *intrinsic* (not primed) RSM model parameters are defined as: . This code was used to generate the ROC curves shown in book Figure 17.1 (a – f) and the *pdfs* shown in book Figure 17.2 (a -f); the displayed plots alternate between ROC curves and pdf plots in the Plots window.

# mainRsmPdfRoc.R

rm(list = ls())

library(RJafroc)

library(ggplot2)

muArr <- c(0.001,1,2,3,4,5);lambda <- 1;nu <- 1; Lmax <- 1

for (i in 1:length(muArr)) {

mu <- muArr[i]

ret1 <- PlotRsmOperatingCharacteristics(

mu, lambda, nu,

type = "ALL", lesionDistribution = c(Lmax,1), legendPosition = "none"

)

pdfPlots <- ret1$PDFPlot +

scale\_color\_manual(values = c("black","darkgrey"), guide = FALSE) +

theme(axis.title.y = element\_text(size = 25,face="bold"),

axis.title.x = element\_text(size = 30,face="bold"),

legend.title = element\_blank(),

legend.position = c(0.25,0.95),

legend.direction = "horizontal", # use for D, E F

legend.text = element\_text(size = 20, face = "bold"),

legend.key.size = unit(2, "lines")) +

scale\_x\_continuous(expand = c(0, 0)) +

scale\_y\_continuous(expand = c(0, 0))

pdfPlots$layers[[1]]$aes\_params$size <- 2 # line

print(pdfPlots)

#next

ROCPlot <- ret1$ROCPlot + scale\_color\_manual(values = "black")

fpfMax <- max(ret1$ROCPlot$data$FPF)

tpfMax <- max(ret1$ROCPlot$data$TPF)

if (fpfMax < 0.99){

fpfCross <- (fpfMax + tpfMax) / 2

tpfCross <- fpfCross

endPoint <- data.frame(FPF = fpfMax, TPF = tpfMax)

ds <- data.frame(FPF = c(fpfMax, fpfCross), TPF = c(tpfMax, tpfCross))

diagonal <- data.frame(FPF = c(0, 1), TPF = c(0, 1))

dsText <- data.frame(FPF = (fpfMax + fpfCross)/2 + 0.05, TPF = (tpfMax + tpfCross)/2)

ROCPlot <- ROCPlot +

geom\_point(data = endPoint,

mapping = aes(x = FPF, y = TPF),

shape = 15, size = 7) +

geom\_line(data = ds,

mapping = aes(x = FPF, y = TPF),

linetype = 2, size = 2) +

geom\_line(data = diagonal,

mapping = aes(x = FPF, y = TPF),

linetype = 2, size = 2) +

geom\_text(data = dsText,

mapping = aes(x = FPF, y = TPF),

label = "d[s]", parse = TRUE, size = 10)

}

ROCPlot <- ROCPlot +

theme(axis.title.y = element\_text(size = 25,face="bold"),

axis.title.x = element\_text(size = 30,face="bold")) +

scale\_x\_continuous(expand = c(0, 0)) +

scale\_y\_continuous(expand = c(0, 0))

ROCPlot$layers[[1]]$aes\_params$size <- 2 # line

ROCPlot$layers[[2]]$aes\_params$size <- 2 # line

print(ROCPlot)

cat("mu = ", mu,

"\nlambda = ", lambda,

"\nnu = ", nu,

"\nAUC = ", ret1$aucROC,

"\nfpfMax = ", fpfMax,

"\ntpfMax = ", tpfMax,"\n")

}

The *pdf* is displayed at line 14-26 while the ROC, with the superposed perpendicular line from the end-point to the chance diagonal, whose length  is proportional to search performance, is displayed at line 29-60.

# Online Appendix 17.E: Is FROC good?

This relates to book section 17.10. The code in mainIsFrocGood.R was used to generate the plots in book Figure 17.7 (a- f), which make the case that the FROC is a poor descriptor of performance.

### Online Appendix 17.E.1: Code listing

# mainIsFrocGood.R

rm(list = ls())

library(RJafroc)

library(ggplot2)

logseq <- function( d1, d2, n) {

logf <- log(d2/d1)/(n-1)

return (exp(seq(log(d1), log(d2), logf)))

}

Lmax <- 1;K2 <- 700;Lk2 <- floor(runif(K2, 1, Lmax + 1))

nLesPerCase <- unique(Lk2)

lesionDist <- array(dim = c(length(nLesPerCase), 2))

for (i in nLesPerCase) lesionDist[i, ] <- c(i, sum(Lk2 == i)/K2)

## PART I

lambda <- 1;nu <- 1

cat("Vary mu only, lambda and nu equal to 1\n")

muArr <- logseq(0.001, 10, 9)

aucRoc <- rep(NA, length(muArr))

aucAfroc <- aucRoc;aucFroc <- aucRoc

for (i in 1:length(muArr)) { # vary mu loop

mu <- muArr[i]

ret1 <- PlotRsmOperatingCharacteristics(

mu, lambda, nu,

lesionDistribution = lesionDist

)

cat("mu = ", mu,", lambda = ", lambda,

", nu = ", nu, ", aucFroc = ", ret1$aucFROC,

", aucRoc = ", ret1$aucROC,", aucAfroc = ", ret1$aucAFROC,"\n")

aucRoc[i] <- ret1$aucROC

aucAfroc[i] <- ret1$aucAFROC

aucFroc[i] <- ret1$aucFROC

}

cat("approx slope AFROC vs ROC =",

(aucAfroc[9]-aucAfroc[1])/(aucRoc[9]-aucRoc[1]),"\n")

aucRocFroc <- data.frame(aucRoc = aucRoc, aucFroc = aucFroc)

aucRocAfroc <- data.frame(aucRoc = aucRoc, aucAfroc = aucAfroc)

muArr2 <- seq(1, 10, by = 0.1)

aucRoc2 <- rep(NA, length(muArr))

aucAfroc2 <- aucRoc2;aucFroc2 <- aucRoc2

for (i in 1:length(muArr2)) { # vary mu loop

mu <- muArr2[i]

ret1 <- PlotRsmOperatingCharacteristics(

mu, lambda, nu,

lesionDistribution = lesionDist

)

aucRoc2[i] <- ret1$aucROC

aucAfroc2[i] <- ret1$aucAFROC

aucFroc2[i] <- ret1$aucFROC

}

plotCurveRocFroc <- data.frame(

aucRoc = c(aucRoc, aucRoc2), aucFroc = c(aucFroc, aucFroc2))

plotRocFroc <- ggplot(

data = aucRocFroc, mapping = aes(x = aucRoc, y = aucFroc)) +

geom\_point(size = 5) +

geom\_line(data = plotCurveRocFroc, size = 2) +

xlab("ROC-AUC") + ylab("FROC-AUC") +

theme(axis.title.y = element\_text(size = 25,face="bold"),

axis.title.x = element\_text(size = 30,face="bold"))

print(plotRocFroc)

plotCurveRocAfroc <- data.frame(

aucRoc = c(aucRoc, aucRoc2), aucAfroc = c(aucAfroc, aucAfroc2))

plotRocAfroc <- ggplot(

data = aucRocAfroc, mapping = aes(x = aucRoc, y = aucAfroc)) +

geom\_point(size = 5) +

geom\_line(data = plotCurveRocAfroc, size = 2) +

xlab("ROC-AUC") + ylab("AFROC-AUC") +

theme(axis.title.y = element\_text(size = 25,face="bold"),

axis.title.x = element\_text(size = 30,face="bold"))

print(plotRocAfroc)

## PART II

mu <- 2;nu <- 1

cat("\nVary lambda only, mu = 2 and nu = 1\n")

lambdaArr <- logseq(0.2, 8, 9)

aucRoc <- rep(NA, length(lambdaArr))

aucAfroc <- aucRoc;aucFroc <- aucRoc

for (i in 1:length(lambdaArr)) {

lambda <- lambdaArr[i]

ret1 <- PlotRsmOperatingCharacteristics(

mu, lambda, nu,

lesionDistribution = lesionDist

)

cat("mu = ", mu,", lambda = ", lambda,

", nu = ", nu, ", aucFroc = ", ret1$aucFROC,

", aucRoc = ", ret1$aucROC,", aucAfroc = ", ret1$aucAFROC,"\n")

aucRoc[i] <- ret1$aucROC

aucAfroc[i] <- ret1$aucAFROC

aucFroc[i] <- ret1$aucFROC

}

cat("approx slope AFROC vs ROC =",

(aucAfroc[9]-aucAfroc[1])/(aucRoc[9]-aucRoc[1]),"\n")

aucRocFroc <- data.frame(aucRoc = aucRoc, aucFroc = aucFroc)

aucRocAfroc <- data.frame(aucRoc = aucRoc, aucAfroc = aucAfroc)

lambdaArr2 <- seq(0.2, 8, by = 0.1)

aucRoc2 <- rep(NA, length(lambdaArr2))

aucAfroc2 <- aucRoc2;aucFroc2 <- aucRoc2

for (i in 1:length(lambdaArr2)) { # vary mu loop

lambda <- lambdaArr2[i]

ret1 <- PlotRsmOperatingCharacteristics(

mu, lambda, nu,

lesionDistribution = lesionDist

)

aucRoc2[i] <- ret1$aucROC

aucAfroc2[i] <- ret1$aucAFROC

aucFroc2[i] <- ret1$aucFROC

}

plotCurveRocFroc <- data.frame(

aucRoc = c(aucRoc, aucRoc2), aucFroc = c(aucFroc, aucFroc2))

plotRocFroc <- ggplot(

data = aucRocFroc, mapping = aes(x = aucRoc, y = aucFroc)) +

geom\_point(size = 5) +

geom\_line(data = plotCurveRocFroc, size = 2) +

xlab("ROC-AUC") + ylab("FROC-AUC") +

theme(axis.title.y = element\_text(size = 25,face="bold"),

axis.title.x = element\_text(size = 30,face="bold"))

print(plotRocFroc)

plotCurveRocAfroc <- data.frame(

aucRoc = c(aucRoc, aucRoc2), aucAfroc = c(aucAfroc, aucAfroc2))

plotRocAfroc <- ggplot(

data = aucRocAfroc, mapping = aes(x = aucRoc, y = aucAfroc)) +

geom\_point(size = 5) +

geom\_line(data = plotCurveRocAfroc, size = 2) +

xlab("ROC-AUC") + ylab("AFROC-AUC") +

theme(axis.title.y =

element\_text(size = 25,face="bold"),

axis.title.x = element\_text(size = 30,face="bold"))

print(plotRocAfroc)

## PART III

mu <- 2;lambda <- 1

cat("\nVary nu only, mu = 2 and lambda = 1\n")

nuArr <- logseq(0.2, 8, 9)

aucRoc <- rep(NA, length(nuArr));aucAfroc <- aucRoc;aucFroc <- aucRoc

for (i in 1:length(nuArr)) {

nu <- nuArr[i]

ret1 <- PlotRsmOperatingCharacteristics(

mu, lambda, nu,

lesionDistribution = lesionDist,

llfRange = c(0,1)

)

cat("mu = ", mu,", lambda = ", lambda,

", nu = ", nu, ", aucFroc = ", ret1$aucFROC,

", aucRoc = ", ret1$aucROC,", aucAfroc = ", ret1$aucAFROC,"\n")

aucRoc[i] <- ret1$aucROC

aucAfroc[i] <- ret1$aucAFROC

aucFroc[i] <- ret1$aucFROC

}

cat("approx slope AFROC vs ROC =",

(aucAfroc[9]-aucAfroc[1])/(aucRoc[9]-aucRoc[1]),"\n")

aucRocFroc <- data.frame(aucRoc = aucRoc, aucFroc = aucFroc)

aucRocAfroc <- data.frame(aucRoc = aucRoc, aucAfroc = aucAfroc)

nuArr2 <- seq(0.2, 8, by = 0.1)

aucRoc2 <- rep(NA, length(nuArr2))

aucAfroc2 <- aucRoc2;aucFroc2 <- aucRoc2

for (i in 1:length(nuArr2)) { # vary mu loop

nu <- nuArr[2]

ret1 <- PlotRsmOperatingCharacteristics(

mu, lambda, nu,

lesionDistribution = lesionDist

)

aucRoc2[i] <- ret1$aucROC

aucAfroc2[i] <- ret1$aucAFROC

aucFroc2[i] <- ret1$aucFROC

}

plotCurveRocFroc <- data.frame(

aucRoc = c(aucRoc, aucRoc2), aucFroc = c(aucFroc, aucFroc2))

plotRocFroc <- ggplot(

data = aucRocFroc, mapping = aes(x = aucRoc, y = aucFroc)) +

geom\_point(size = 5) +

geom\_line(data = plotCurveRocFroc, size = 2) +

xlab("ROC-AUC") + ylab("FROC-AUC") +

theme(axis.title.y = element\_text(size = 25,face="bold"),

axis.title.x = element\_text(size = 30,face="bold"))

print(plotRocFroc)

plotCurveRocAfroc <- data.frame(

aucRoc = c(aucRoc, aucRoc2), aucAfroc = c(aucAfroc, aucAfroc2))

plotRocAfroc <- ggplot(

data = aucRocAfroc, mapping = aes(x = aucRoc, y = aucAfroc)) +

geom\_point(size = 5) +

geom\_line(data = plotCurveRocAfroc, size = 2) +

xlab("ROC-AUC") + ylab("AFROC-AUC") +

theme(axis.title.y = element\_text(size = 25,face="bold"),

axis.title.x = element\_text(size = 30,face="bold"))

print(plotRocAfroc)

The in-line (i.e., temporary, visible only in the sourced file) defined function logseq() creates an array with equally spaced values on a logarithmic scale, used at lines 18, 92 and 161. The code is divided into 3 parts:

* Part I, between lines 16 and 73, calculates ,  and  for varying , values specified at line 19, on a logarithmic scale, and at line 40, on a linear scale, with . The two scales are used to generate 18 points giving the appearance of a continuous curve.
* Part II, between lines 75 and 133, calculates the same AUCs for varying , values specified at line 78 and 99, with .
* Part III, between lines 135 and 191, calculates them for varying , values specified at line 138 and 159, with .

Source the code to get the following output, summarized in book Figure 17.7:

### Online Appendix 17.E.2: Code listing

> source(...)

Vary mu only, lambda and nu equal to 1

mu = 0.001 , lambda = 1 , nu = 1 , aucFroc = 0.4986837 , aucRoc = 0.5000003 , aucAfroc = 1.002317e-06

mu = 0.003162278 , lambda = 1 , nu = 1 , aucFroc = 0.498754 , aucRoc = 0.5000025 , aucAfroc = 1.007167e-05

mu = 0.01 , lambda = 1 , nu = 1 , aucFroc = 0.4989729 , aucRoc = 0.5000256 , aucAfroc = 0.0001019791

mu = 0.03162278 , lambda = 1 , nu = 1 , aucFroc = 0.4996293 , aucRoc = 0.5002648 , aucAfroc = 0.001050011

mu = 0.1 , lambda = 1 , nu = 1 , aucFroc = 0.5013515 , aucRoc = 0.5028535 , aucAfroc = 0.01104766

mu = 0.3162278 , lambda = 1 , nu = 1 , aucFroc = 0.5033454 , aucRoc = 0.5308322 , aucAfroc = 0.1193661

mu = 1 , lambda = 1 , nu = 1 , aucFroc = 0.4797164 , aucRoc = 0.7148289 , aucAfroc = 0.577889

mu = 3.162278 , lambda = 1 , nu = 1 , aucFroc = 0.2985952 , aucRoc = 0.9751592 , aucAfroc = 0.9693428

mu = 10 , lambda = 1 , nu = 1 , aucFroc = 0.09986048 , aucRoc = 0.9999773 , aucAfroc = 0.9999751

approx slope AFROC vs ROC = 2.00004

Vary lambda only, mu = 2 and nu = 1

mu = 2 , lambda = 0.2 , nu = 1 , aucFroc = 0.07954922 , aucRoc = 0.9257254 , aucAfroc = 0.9191906

mu = 2 , lambda = 0.3171666 , nu = 1 , aucFroc = 0.1261518 , aucRoc = 0.9220264 , aucAfroc = 0.9118694

mu = 2 , lambda = 0.5029734 , nu = 1 , aucFroc = 0.2000557 , aucRoc = 0.9164018 , aucAfroc = 0.9007912

mu = 2 , lambda = 0.7976319 , nu = 1 , aucFroc = 0.3172549 , aucRoc = 0.9080394 , aucAfroc = 0.8844514

mu = 2 , lambda = 1.264911 , nu = 1 , aucFroc = 0.5031134 , aucRoc = 0.8960075 , aucAfroc = 0.8612441

mu = 2 , lambda = 2.005938 , nu = 1 , aucFroc = 0.7978538 , aucRoc = 0.8794642 , aucAfroc = 0.8299777

mu = 2 , lambda = 3.181083 , nu = 1 , aucFroc = 1.265263 , aucRoc = 0.8580084 , aucAfroc = 0.7905774

mu = 2 , lambda = 5.044667 , nu = 1 , aucFroc = 2.006496 , aucRoc = 0.831995 , aucAfroc = 0.7442658

mu = 2 , lambda = 8 , nu = 1 , aucFroc = 3.181968 , aucRoc = 0.8025288 , aucAfroc = 0.6924404

approx slope AFROC vs ROC = 1.840556

Vary nu only, mu = 2 and lambda = 1

mu = 2 , lambda = 1 , nu = 0.2 , aucFroc = 0.1516529 , aucRoc = 0.653525 , aucAfroc = 0.5208837

mu = 2 , lambda = 1 , nu = 0.3171666 , aucFroc = 0.2160672 , aucRoc = 0.7187345 , aucAfroc = 0.6133168

mu = 2 , lambda = 1 , nu = 0.5029734 , aucFroc = 0.291779 , aucRoc = 0.7953809 , aucAfroc = 0.7219614

mu = 2 , lambda = 1 , nu = 0.7976319 , aucFroc = 0.3666869 , aucRoc = 0.8712134 , aucAfroc = 0.8294525

mu = 2 , lambda = 1 , nu = 1.264911 , aucFroc = 0.4233506 , aucRoc = 0.9285766 , aucAfroc = 0.9107636

mu = 2 , lambda = 1 , nu = 2.005938 , aucFroc = 0.4516746 , aucRoc = 0.9572502 , aucAfroc = 0.9514078

mu = 2 , lambda = 1 , nu = 3.181083 , aucFroc = 0.4592065 , aucRoc = 0.9648751 , aucAfroc = 0.962216

mu = 2 , lambda = 1 , nu = 5.044667 , aucFroc = 0.4599812 , aucRoc = 0.9656594 , aucAfroc = 0.9633276

mu = 2 , lambda = 1 , nu = 8 , aucFroc = 0.4600003 , aucRoc = 0.9656787 , aucAfroc = 0.963355

approx slope AFROC vs ROC = 1.417479

To understand what is going on is to place a breakpoint on an early line, e.g., line 22, and click Source, and then use Next to step line by line through the code. When inside a for-loop, click "get out of loop" to speed up the process. For example, line 24 – 27 executes PlotRsmOperatingCharacteristics() and saves the result to ret1, which is a list containing ret1$aucROC, the area under the RSM predicted ROC curve, ret1$aucAFROC, the area under the RSM predicted AFROC curve and ret1$aucFROC, the area under the RSM predicted FROC curve. Line 35 – 36 calculates the approximate slope of the plot, assumed linear, of AFROC AUC vs. ROC AUC.

# Online Appendix 17.F: Binormal parameters for RSM-generated ROC datasets

This code is not referred to in the book. It shows RSM predicted and binormal model fitted curves to datasets generated by the RSM.

## Online Appendix 17.F.1: Code Listing

# mainRsmVsRocfitR.R

rm(list = ls())

library(RJafroc)

library(ggplot2)

library("numDeriv")

source("Transforms.R")

source("LL.R")

source("RocOperatingPointsFromRatingsTable.R")

source("VarianceAz.R");

source("ChisqrGoodnessOfFit.R")

source("RocfitR.R")

source("AucsRsm.R")

source("PlotRSMBM.R")

lambda <- 1;zeta1 <- -1;nBins <- 5;K1 <- 500;K2 <- 700

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

muArr <- c(2,3);nuArr <- c(0.15,0.25);

cat("lambda = ", lambda, ", zeta1 = ", zeta1, "\n")

seedArr <- c(2,3)

for (s in 1:2){

seed <- seedArr[s];set.seed(seed)

cat("seed = ", seed, "\n")

Lmax <- 1;Lk2 <- floor(runif(K2, 1, Lmax + 1))

nLesPerCase <- unique(Lk2)

lesionDist <- array(dim = c(length(nLesPerCase), 2))

for (i in nLesPerCase) lesionDist[i, ] <- c(i, sum(Lk2 == i)/K2)

cat("Lmax = ", Lmax, "\n")

for (i in 1:2){

for (j in 1:2){

mu <- muArr[i];nu <- nuArr[j]

RowString <- toString(c(seed,mu,nu))

while(1){

frocDataRaw <- SimulateFrocDataset (

mu, lambda, nu,

I = 2, J = 2, K1 = K1, K2,

lesionNum = Lk2, zeta1 = zeta1)

rocDataRaw <- DfFroc2Roc(frocDataRaw)

rocDataBinned <- DfBinDataset(

rocDataRaw,

desiredNumBins = nBins,

opChType = "ROC")

if (length(unique(rocDataBinned$LL[i,j,,1])) != nBins) next

if (length(unique(rocDataBinned$NL[i,j,1:K1,1])) != nBins) next

break

}

rocDataTable <- array(dim = c(2,nBins))

rocDataTable[1,] <- table(rocDataBinned$NL[i,j,1:K1,1])

rocDataTable[2,] <- table(rocDataBinned$LL[i,j,1:K2,1])

retRocfit <- RocfitR(rocDataTable)

if (length(retRocfit) != 6) stop("rocfit failed")

aucs <- AucsRsm(mu = mu,

lambda = lambda,

nu = nu,

lesionDist = lesionDist)

cat("mu=", mu,

",nu=", nu,

",RSM-ROC-AUC = ", aucs$aucROC,

",Az=", retRocfit$Az,

",a=", retRocfit$a,

",b =", retRocfit$b, "\n")

compPlot <- PlotRSMBM(

retRocfit$a,

retRocfit$b,

mu,

lambda,

nu, lesionDist, RowString)

print(compPlot)

next

}

}

}

The code generates 8 plots corresponding to two values of seed, two values of mu and two values of nu. RSM parameters that do not change are listed in line 15: lambda = 1, zeta1 = -1, nBins = 5, K1 = 500 and K2= 700. Line 21 initializes the seed variable (seed = 2 or 3). Changing seed is equivalent to sampling a fresh dataset. Line 31 initializes mu (mu = 2 or 3) and nu (nu = 0.15 or 0.25). These are the intrinsic parameters of the RSM, not the primed values. FROC data is simulated at line 35-38 and converted to ROC data at line 39. A large numbers of cases was deliberately chosen both to minimize sampling variability and to give the binormal model a chance of succeeding; with too few cases the binning method (lines 41 - 44) may not find 5 bins; a check is made at line 45-46 for too few bins. Lines 50 – 52 constructs the ROC data table, the analog of book Table 4.1. The binormal model fitting function, RocfitR(), is called at line 54, followed by a check to see if the algorithm converged, line 55. The binormal model fitting part of the code should be familiar from book Chapter 6. Line 57-60 calls the function AucsRsm() which numerically integrates the ROC and AFROC curves, including any applicable straight line extensions. Lines 67-73 displays the combined ROC plots shown in Figure 1. Source the code to obtain the following output and Figure 1.

> source(...)

K1 = 500 , K2 = 700

lambda = 1 , zeta1 = -1

seed = 2

Lmax = 1

mu= 2 ,nu= 0.15 ,RSM-ROC-AUC = 0.6206955 ,Az= 0.6007239 ,a= 0.2984692 ,b = 0.5870132

mu= 2 ,nu= 0.25 ,RSM-ROC-AUC = 0.6832303 ,Az= 0.6935744 ,a= 0.5885988 ,b = 0.6697229

mu= 3 ,nu= 0.15 ,RSM-ROC-AUC = 0.67924 ,Az= 0.5807186 ,a= 0.2647179 ,b = 0.352378

mu= 3 ,nu= 0.25 ,RSM-ROC-AUC = 0.7609834 ,Az= 0.7323561 ,a= 0.6327625 ,b = 0.2953327

seed = 3

Lmax = 1

mu= 2 ,nu= 0.15 ,RSM-ROC-AUC = 0.6206955 ,Az= 0.6046442 ,a= 0.3321193 ,b = 0.599404

mu= 2 ,nu= 0.25 ,RSM-ROC-AUC = 0.6832303 ,Az= 0.6748706 ,a= 0.5205419 ,b = 0.5245174

mu= 3 ,nu= 0.15 ,RSM-ROC-AUC = 0.67924 ,Az= 0.5912285 ,a= 0.2897353 ,b = 0.3562369

mu= 3 ,nu= 0.25 ,RSM-ROC-AUC = 0.7609834 ,Az= 0.7449772 ,a= 0.5526823 ,b = 0.2914531

All RSM predicted curves are proper, while all binormal fitted curves are improper (all b-values are less than unity). The RSM predicted curves are unaffected by sampling variability, while the binormal fitted curves are affected. Due to the large number of cases, sampling variability is expected to be small, as can be appreciated by comparing Az values for sets of parameters that differ only in seed values (the differences are in the second decimal). In general, the RSM predicted AUCs exceed the binormal fitted values.

To gain a better understanding, insert a break point at line 54 and click Source. Highlight rocDataTable and click Run, yielding the following output:

Browse[2]> rocDataTable

[,1] [,2] [,3] [,4] [,5]

[1,] 363 62 50 15 10

[2,] 365 72 84 52 127

This is the ROC counts table corresponding to data generated for seed = 2, mu = 2 and nu = 0.15. Click Next, highlight retRocFit and click Run:

Browse[2]> retRocfit

$Az

[1] 0.6007239

$StdAz

[1] 0.02433877

$a

[1] 0.2984692

$b

[1] 0.5870132

$zeta

[1] 0.600383 1.042655 1.632115 2.058449

$Cov

[,1] [,2] [,3] [,4] [,5] [,6]

[1,] 0.0061043957 0.0032235948 0.0025683173 0.0014532283 -0.0003421017 -0.001959046

[2,] 0.0032235948 0.0034064352 0.0007971401 -0.0004975542 -0.0026680681 -0.004626343

[3,] 0.0025683173 0.0007971401 0.0035745732 0.0028833998 0.0021310774 0.001612049

[4,] 0.0014532283 -0.0004975542 0.0028833998 0.0039815959 0.0038835526 0.004026398

[5,] -0.0003421017 -0.0026680681 0.0021310774 0.0038835526 0.0074641110 0.008641023

[6,] -0.0019590461 -0.0046263428 0.0016120493 0.0040263979 0.0086410228 0.013440769

This lists the results of the binormal model analysis (Az, StdAz, a, b, zeta and Cov, the last being the covariance matrix. The four values of zeta correspond to nBins = 5.

Click Next twice, highlight aucs and click Run.

Browse[2]> aucs

$aucROC

[1] 0.6206955

$aucAFROC

[1] 0.4743486

In Figure 1, three numbers label each plot, near the top-left corner: seed, mu and nu. The red line is the RSM prediction and the blue line is the binormal model fit.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

Figure : It shows RSM predicted and binormal model fitted curves to datasets generated by the RSM.

## Online Appendix 17.F.2: Comparing RSM prediction to binormal fits

This is related to book section 17.11.2. The aim is to show that the binormal model is a special case of the RSM, or stated more precisely, for not too large datasets generated by the RSM, the binormal model-fitting algorithm behaves as if the data originated from a binormal model simulator. The philosophy is similar to work3 by Hanley showing that the binormal model is quite robust with respect to deviations of the underlying model from strict bi-normality.

The code mainRsmVsEng.R simulates FROC data using the RSM, converts it to highest rating ROC data, bins the data into 5 bins and prints out two sets of 5 integers, the bin counts in non-diseased and diseased cases respectively, the analog of the ROC counts data in book Table 4.1. These were analyzed by the Eng Java program4, see book Section 6.2.7, which yields the binormal parameter values *a*, *b* and the goodness of fit statistic and a p-value, a very small value implies the fit is of poor statistical quality (Google the "Eng ROC software" website, ensure one has the latest version of Java and it is enabled). The binormal fitted (*a,* *b*) parameters were transferred to the appropriate locations between lines 57 - 65.

# mainRsmVsEng.R

rm(list = ls())

library(RJafroc)

library(ggplot2)

library(binom)

source("AucsRsm.R");source("FpTp2FpfTpf.R")

source("PlotBMErrBar.R")

K1 <- 500;K2 <- 700

for (Row in 1:8) {

switch(Row,

"1" = {seed <- 1;set.seed(seed);Lmax <- 1;mu <- 2.0;lambda <- 10;nu <- 1;zeta1 <- -1}, # Row 1

"2" = {seed <- 1;set.seed(seed);Lmax <- 1;mu <- 2.5;lambda <- 10;nu <- 1;zeta1 <- -1}, # Row 2

"3" = {seed <- 1;set.seed(seed);Lmax <- 1;mu <- 3.0;lambda <- 10;nu <- 1;zeta1 <- -1}, # Row 3

"4" = {seed <- 2;set.seed(seed);Lmax <- 1;mu <- 2.5;lambda <- 10;nu <- 1;zeta1 <- -1}, # Row 4

"5" = {seed <- 2;set.seed(seed);Lmax <- 2;mu <- 2.0;lambda <- 10;nu <- 1;zeta1 <- -1},# Row 5

"6" = {seed <- 2;set.seed(seed);Lmax <- 2;mu <- 2.5;lambda <- 10;nu <- 1;zeta1 <- -1},# Row 6

"7" = {seed <- 2;set.seed(seed);Lmax <- 2;mu <- 3.0;lambda <- 10;nu <- 1;zeta1 <- -1},# Row 7

"8" = {seed <- 2;K1 <- 5000;K2 <- 7000;set.seed(seed);Lmax <- 2;mu <- 3.0;lambda <- 1;nu <- 1;zeta1 <- -1},# Row 8

"9" = {seed <- 2;K1 <- 5000;K2 <- 7000;set.seed(seed);Lmax <- 2;mu <- 3.0;lambda <- 0.1;nu <- 1;zeta1 <- -1}# Row 9

)

cat("K1 = ", K1,

"\nK2 = ", K2,

"\nzeta1 = ", zeta1,

"\nseed = ", seed,

"\nLmax = ", Lmax,

"\nmu = ", mu,

"\nlambda = ", lambda,

"\nnu = ", nu, "\n")

Lk2 <- floor(runif(K2, 1, Lmax + 1))

nLesPerCase <- unique(Lk2);lesionDist <- array(dim = c(length(nLesPerCase), 2))

for (i in nLesPerCase) lesionDist[i, ] <- c(i, sum(Lk2 == i)/K2)

frocDataRaw <- SimulateFrocDataset(mu, lambda, nu, I = 1, J = 1, K1, K2, lesionNum = Lk2, zeta1 = zeta1)

rocDataRaw <- DfFroc2Roc(frocDataRaw)

rocDataBinned <- DfBinDataset(rocDataRaw, desiredNumBins = 5, opChType = "ROC")

fp <-rocDataBinned$NL[1,1,1:K1,1];tp <- rocDataBinned$LL[1,1,1:K2,1]

ret1 <- FpTp2FpfTpf(fp, tp)

fpCountsOrg <- ret1$fpCounts;tpCountsOrg <- ret1$tpCounts;fpf <- ret1$fpf;tpf <- ret1$tpf;zetas <- ret1$zetas

nBins <- length(fpCountsOrg)

rocDataTable <- array(dim = c(2,nBins))

rocDataTable[1,] <- fpCountsOrg;rocDataTable[2,] <- tpCountsOrg

aucs <- AucsRsm(mu = mu, lambda = lambda, nu = nu, lesionDist = lesionDist)

cat("RSM-ROC-AUC = ", aucs$aucROC, "\n")

print(rocDataTable)

next

# copy the last two rows of output to Eng program; delete bracket stuff leaving numbers only with spaces; select format 3

# Run Program

# compare to Table 1 in online appendix

switch(Row,

# the following values were transferred from the Eng program output after analyzing data generated

# by mainRsmVsEng.R using the appropriate value of Row

"1" = {a <- 1.0066;b <- 0.8182}, # Row 1

"2" = {a <- 1.5134;b <- 0.7617}, # Row 2

"3" = {a <- 1.9561;b <- 0.7643}, # Row 3

"4" = {a <- 1.2324;b <- 0.7078}, # Row 4

"5" = {a <- 1.3246;b <- 0.8715},# Row 5

"6" = {a <- 1.5939;b <- 0.7325},# Row 6

"7" = {a <- 2.1235;b <- 0.74086},# Row 7

"8" = {a <- 2.4329;b <- 0.4719},# Row 8

"9" = {a <- NA;b <- NA} # Row 9

)

print(PlotBMErrBar(a, b, rocDataTable, Row))

}

The switch() statement at line 11 allows selection of a specific set of parameters depending on the value of Row specified at line 10. For example, if Row = 1, i.e., on the first iteration of the for-loop, then the set of parameters on line 12 are selected, if Row = 2, then the set of parameters on line 13 are selected, etc. The remaining parameters of the RSM are  and . The FROC data is simulated at line 35 and the next line converts it to ROC data. Line 38 bins the data appropriately for ROC analysis. Lines 39 – 44 constructs the counts table. The numbers of cases is intentionally large to minimize sampling variability and to allow 5 bins to be found, where each bin has at least 5 counts in both TP and FP categories (search RJafroc documentation for BinDataset()for details). Note that these values are over-ridden by even larger values for Row = 8 and 9. Ensure that line 51 is not commented, and click Source. The output was used to populate Table 1. Each data table was copied to the Eng website, the Java version of ROCFIT was run, and the output used to populate Table 2 and the entries in line 59 – 67.

Table 1: This table lists the ROC counts table corresponding to each value of Row. Each counts table was analyzed by the Eng Java program, yielding the values in Table 2 and the a and b parameters were copied to lines 57-65.

|  |  |  |  |
| --- | --- | --- | --- |
| Row | K1 | K2 | rocDataTable |
| 1 | 500 | 700 | [,1] [,2] [,3] [,4] [,5]  [1,] 201 140 106 34 19  [2,] 75 124 158 98 245 |
| 2 | [,1] [,2] [,3] [,4] [,5]  [1,] 151 213 112 17 7  [2,] 21 79 180 129 291 |
| 3 | [,1] [,2] [,3] [,4] [,5]  [1,] 302 104 62 25 7  [2,] 28 41 83 120 428 |
| 4 | [,1] [,2] [,3] [,4] [,5]  [1,] 240 174 51 23 12  [2,] 74 120 96 124 286 |
| 5 | [,1] [,2] [,3] [,4] [,5]  [1,] 133 194 80 74 19  [2,] 20 104 69 224 283 |
| 6 | [,1] [,2] [,3] [,4] [,5]  [1,] 148 225 71 51 5  [2,] 19 69 76 243 293 |
| 7 | [,1] [,2] [,3] [,4] [,5]  [1,] 303 120 56 15 6  [2,] 24 25 89 130 432 |
| 8 | 5000 | 7000 | [,1] [,2] [,3] [,4] [,5]  [1,] 3780 903 291 21 5  [2,] 159 104 716 986 5035 |
| 9 | [,1] [,2] [,3] [,4] [,5]  [1,] 4843 155 2 0 0  [2,] 190 715 868 870 4357 |

Table 2: This table shows the results of simulating ROC ratings tables using the RSM for parameter values specified in columns 2 – 6 and fitting each ratings table using the binormal model. The corresponding binormal model fitted ROC curves are shown in Figure 2: ROC plot produced for the code listed above.. For each row the number of non-diseased cases was 500, the number of diseased cases was 700, and the reporting threshold was set to -1. The resulting ratings table, e.g., §17.11.2.1, was analyzed by Eng Java software to obtain the values listed in the remaining columns. Note the close correspondence between the RSM-AUC and the binormal fitted AUC, with the former being slightly larger; this has to do with the proper vs. improper nature of RSM and binormal model fits; proper fits are expected to have a larger AUC. The last column lists the p-value for the chi-square goodness of fit statistic. Values greater than 0.001 are generally considered good fits. [NA: the chi-square goodness of fit statistics could not be calculated because some of the cell counts were less than five or data was degenerate.]

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | *R code* | | | | | | | ***Eng Java program*** | | | | | |
| ***Row*** | *K1* | *K2* | *seed* |  |  |  |  |  | ***a*** | ***b*** |  |  | ***df*** | ***p-value*** |
| 1 | 500 | 700 | 1 | 1 | 2.0 | 10 | 1 | 0.7875 | 1.0066 | 0.8182 | 0.7820 | 2.3751 | 2 | 0.3050 |
| 2 | 1 | 1 | 2.5 | 10 | 1 | 0.879 | 1.5134 | 0.7617 | 0.8857 | 0.5178 | 2 | 0.7719 |
| 3 | 1 | 1 | 3.0 | 10 | 1 | 0.938 | 1.9561 | 0.7643 | 0.9399 | 0.1148 | 2 | 0.9442 |
| 4 | 2 | 1 | 2.5 | 10 | 1 | 0.879 | 1.2324 | 0.7078 | 0.8428 | 1.5633 | 2 | 0.4577 |
| 5 | 2 | 2 | 2.0 | 10 | 1 | 0.841 | 1.3246 | 0.8715 | 0.8410 | 3.7776 | 2 | 0.1513 |
| 6 | 2 | 2 | 2.5 | 10 | 1 | 0.920 | 1.5939 | 0.7325 | 0.9007 | 1.7056 | 2 | 0.4262 |
| 7 | 2 | 2 | 3.0 | 10 | 1 | 0.963 | 2.1235 | 0.7408 | 0.9560 | 8.4548 | 2 | 0.0146 |
| 8 | 5000 | 7000 | 2 | 2 | 3.0 | 1 | 1 | 0.984 | 2.4329 | 0.4924 | 0.9855 | 100.75 | 2 | 0.0000 |
| 9 | 2 | 2 | 3.0 | 0.1 | 1 | 0.987 | NA | NA | 1 | NA | | |

Now, comment line 51 and click Source, yielding Figure 2.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  | NA |

Figure 2: These plots show RSM-generated ROC operating points using parameters and seeds specified in Online Appendix 17.F.2 and the corresponding binormal model fitted curves. The integer label at the top of each plot corresponds to a value of Row in Online Appendix 17.F.2. The (*a*, *b*) parameters were obtained by running lines 1 - 51 in mainRsmVsEng.R, transferring the ROC counts table data to the Eng Java program, running the Eng program and transferring the parameter values to the appropriate location, between lines 59 - 67. Once all values are populated, the plots were obtained by sourcing the file with different values for variable Row, in the integer range 1 to 8, specified at line 10. Even with the large number of cases, sampling variability affects the binormal model fits: e.g., the binormal model curves in plots labeled "2" and "4" differ only in seed values. These plots show that over a wide range of parameters, RSM generated ROC data is fitted reasonably by the binormal model. As far as the binormal model-fitting software is concerned, the counts data arose from two normal distributions. The only exception is the figure for Row = 8, where the data is visually fit by the binormal model but fails the chi-square test. This reflects Hanley's findings that detecting deviations from an underlying strictly binormal model requires very large numbers of cases (12,000 in the current example). For Row = 9, the binned data is degenerate and cannot be uniquely fitted by the binormal model, although it can be fitted by methods described in book Chapters 19 and 20. *Since the binormal model has been used successfully for over three decades, the ability to the RSM to mimic it is an important justification for the validity of the RSM.*

# Online Appendix 17.G: Explanations of Swets et al observations

The following code shows that the RSM is consistent with the observations made by Swets et al3 that the b-parameter decreases with increasing lesion contrast and the ratio is approximately constant for a fixed set of experimental conditions. The results of sourcing this code were used to populate book Table 17.3. To maintain a fixed set of experimental conditions, whenever a parameter is varied, the remaining two parameters are adjusted to maintain RSM ROC-AUC fixed at the value specified in line 16. This is currently set to 0.7 but Online Appendix 17.G.2 contains results for this value and AUC = 0.8.

## Online Appendix 17.G.1: Code Listing

# mainRsmSwetsObservations.R

rm(list = ls())

library(RJafroc)

source("rsmPdfMeansAndStddevs.R")

source("FindParamFixAuc.R")

logseq <- function( d1, d2, n) {

logf <- log(d2/d1)/(n-1)

return (exp(seq(log(d1), log(d2), logf)))

}

Lmax <- 1;K2 <- 700;Lk2 <- floor(runif(K2, 1, Lmax + 1)) # K2 is only used to get an accurate lesion distribution vector

nLesPerCase <- unique(Lk2);lesionDist <- array(dim = c(length(nLesPerCase), 2))

for (i in nLesPerCase) lesionDist[i, ] <- c(i, sum(Lk2 == i)/K2)

RsmRocAuc <- 0.7 # parameters adjusted to attain this value; 0.7 or 0.8

cat("RsmRocAuc constraint = ", RsmRocAuc, "\n")

# Part A

lambda <- 2;cat("\nVary mu and nu only, ", "lambda = ", lambda, "\n")

muArr <- logseq(2,5,10)

for (i in 1:length(muArr)) {

mu <- muArr[i];nu <- NA # intrinsic parameters

retParms <- FindParamFixAuc(mu, lambda, nu, lesionDist, RsmRocAuc)

if (!is.na(retParms)) nu <- retParms else next

ret <- rsmPdfMeansAndStddevs(mu, lambda, nu, lesionDist)

meanN <- ret$meanN;meanD <- ret$meanD;stdDevN <- ret$stdErrN;stdDevD <- ret$stdErrD

ret1 <- RsmOperatingCharacteristics(mu, lambda, nu, type = "ROC", lesionDistribution = lesionDist, legendPosition = "none")

cat("mu = ", mu,", lambda = ", lambda,

", nu = ", nu, ", AUC = ", ret1$aucROC,

", bParm = ", stdDevN/stdDevD, ", dmu/dsigma = ", (meanD - meanN)/(stdDevD - stdDevN), "\n")

next

}

# Part B

nu <- 1;cat("\nVary mu and lambda only, ", "nu = ", nu, "\n")

lambdaArr <- logseq(1, 5, 10)

for (i in 1:length(lambdaArr)) {

lambda <- lambdaArr[i]; mu <- NA; # intrinsic parameters

retParms <- FindParamFixAuc(mu, lambda, nu, lesionDist, RsmRocAuc)

if (!is.na(retParms)) mu <- retParms else next

ret <- rsmPdfMeansAndStddevs(mu, lambda, nu, lesionDist)

meanN <- ret$meanN;meanD <- ret$meanD;stdDevN <- ret$stdErrN;stdDevD <- ret$stdErrD

ret1 <- RsmOperatingCharacteristics(mu, lambda, nu, type = "ROC", lesionDistribution = lesionDist, legendPosition = "none")

cat("mu = ", mu,", lambda = ", lambda,

", nu = ", nu, ", AUC = ", ret1$aucROC,

", bParm = ", stdDevN/stdDevD, ", dmu/dsigma = ", (meanD - meanN)/(stdDevD - stdDevN), "\n")

next

}

# Part C

mu <- 2;cat("\nVary lambda and nu only,", "mu = ", mu, "\n")

# nuArr <- logseq(1, 10, 10)

lambdaArr <- logseq(0.1, 5, 10)

for (i in 1:length(lambdaArr)) {

lambda <- lambdaArr[i]; nu <- NA; # intrinsic parameters

# retParms <- FindParamFixAuc(mu, lambda, nu, lesionDist, RsmRocAuc)

# if (!is.na(retParms)) lambda <- retParms else next

retParms <- FindParamFixAuc(mu, lambda, nu, lesionDist, RsmRocAuc)

if (!is.na(retParms)) nu <- retParms else next

ret <- rsmPdfMeansAndStddevs(mu, lambda, nu, lesionDist)

meanN <- ret$meanN;meanD <- ret$meanD;stdDevN <- ret$stdErrN;stdDevD <- ret$stdErrD

ret1 <- RsmOperatingCharacteristics(mu, lambda, nu, type = "ROC", lesionDistribution = lesionDist, legendPosition = "none")

cat("mu = ", mu,", lambda = ", lambda,

", nu = ", nu, ", AUC = ", ret1$aucROC,

", bParm = ", stdDevN/stdDevD, ", dmu/dsigma = ", (meanD - meanN)/(stdDevD - stdDevN), "\n")

next

}

Like that table that it populates, the code is organized into three parts, A, B and C. Part A, lines 19 – 35, varies  for constant AUC for . Part B, lines 36 – 53, varies  for constant AUC for . Part C, lines 54 – 73, varies  for constant AUC for . The function FindParamFixAuc(), line 24, finds the missing parameter, indicated by initializing it with NA, prior to the function call, given the two other parameters of the RSM. The function rsmPdfMeansAndStddevs(), line 26, calculates the means and standard deviations of the two distributions, after appropriately normalizing. Line 32 prints out the b-parameter and . A seed variable is not needed in this function, as random sampling is not involved. Source the code to obtain the following output, which was used to populate book Table 17.3.

## Online Appendix 17.G.2: Code Output

> source(...)

RsmRocAuc constraint = 0.7

Vary mu and nu only, lambda = 2

mu = 2 , lambda = 2 , nu = 0.3040194 , AUC = 0.6999999 , bParm = 0.7539348 , dmu/dsigma = 3.136005

mu = 2.214346 , lambda = 2 , nu = 0.2600273 , AUC = 0.7000013 , bParm = 0.7107209 , dmu/dsigma = 2.812443

mu = 2.451665 , lambda = 2 , nu = 0.2252662 , AUC = 0.7000022 , bParm = 0.66705 , dmu/dsigma = 2.585494

mu = 2.714418 , lambda = 2 , nu = 0.1973748 , AUC = 0.700004 , bParm = 0.6235964 , dmu/dsigma = 2.425911

mu = 3.00533 , lambda = 2 , nu = 0.1746062 , AUC = 0.7000075 , bParm = 0.5810496 , dmu/dsigma = 2.315364

mu = 3.327421 , lambda = 2 , nu = 0.1556552 , AUC = 0.7000147 , bParm = 0.5400178 , dmu/dsigma = 2.241479

mu = 3.684031 , lambda = 2 , nu = 0.1395508 , AUC = 0.7000293 , bParm = 0.5009616 , dmu/dsigma = 2.195411

mu = 4.078861 , lambda = 2 , nu = 0.1255604 , AUC = 0.7000218 , bParm = 0.4641586 , dmu/dsigma = 2.170306

mu = 4.516005 , lambda = 2 , nu = 0.1132283 , AUC = 0.7000323 , bParm = 0.429762 , dmu/dsigma = 2.16181

mu = 5 , lambda = 2 , nu = 0.102157 , AUC = 0.6999561 , bParm = 0.3977542 , dmu/dsigma = 2.165567

Vary mu and lambda only, nu = 1

mu = 0.94534 , lambda = 1 , nu = 1 , AUC = 0.6999937 , bParm = 0.9767467 , dmu/dsigma = 25.43453

mu = 0.9900899 , lambda = 1.195813 , nu = 1 , AUC = 0.699996 , bParm = 0.9763296 , dmu/dsigma = 25.60972

mu = 1.036822 , lambda = 1.429969 , nu = 1 , AUC = 0.6999951 , bParm = 0.9759534 , dmu/dsigma = 25.83105

mu = 1.085303 , lambda = 1.709976 , nu = 1 , AUC = 0.6999922 , bParm = 0.9755036 , dmu/dsigma = 25.96241

mu = 1.135354 , lambda = 2.044812 , nu = 1 , AUC = 0.7000048 , bParm = 0.9747737 , dmu/dsigma = 25.78277

mu = 1.18661 , lambda = 2.445213 , nu = 1 , AUC = 0.7000022 , bParm = 0.9734671 , dmu/dsigma = 25.02159

mu = 1.238898 , lambda = 2.924018 , nu = 1 , AUC = 0.7000007 , bParm = 0.9711321 , dmu/dsigma = 23.42021

mu = 1.292001 , lambda = 3.496579 , nu = 1 , AUC = 0.7000001 , bParm = 0.9671901 , dmu/dsigma = 20.92522

mu = 1.345696 , lambda = 4.181255 , nu = 1 , AUC = 0.6999934 , bParm = 0.9609738 , dmu/dsigma = 17.80463

mu = 1.399901 , lambda = 5 , nu = 1 , AUC = 0.7000011 , bParm = 0.9518188 , dmu/dsigma = 14.54334

Vary lambda and nu only, mu = 2

mu = 2 , lambda = 0.1 , nu = 0.258024 , AUC = 0.7000003 , bParm = 0.8966579 , dmu/dsigma = 16.09127

mu = 2 , lambda = 0.1544452 , nu = 0.2594364 , AUC = 0.7000003 , bParm = 0.8631606 , dmu/dsigma = 11.27213

mu = 2 , lambda = 0.2385332 , nu = 0.2616054 , AUC = 0.7000003 , bParm = 0.8274848 , dmu/dsigma = 8.132437

mu = 2 , lambda = 0.3684031 , nu = 0.2649264 , AUC = 0.7000003 , bParm = 0.7940112 , dmu/dsigma = 6.083536

mu = 2 , lambda = 0.568981 , nu = 0.2699885 , AUC = 0.7000004 , bParm = 0.7675247 , dmu/dsigma = 4.750151

mu = 2 , lambda = 0.8787639 , nu = 0.2776535 , AUC = 0.7000005 , bParm = 0.7516999 , dmu/dsigma = 3.896845

mu = 2 , lambda = 1.357209 , nu = 0.2891525 , AUC = 0.7000006 , bParm = 0.7479598 , dmu/dsigma = 3.380376

mu = 2 , lambda = 2.096144 , nu = 0.3061914 , AUC = 0.6999999 , bParm = 0.7551508 , dmu/dsigma = 3.11838

mu = 2 , lambda = 3.237394 , nu = 0.3310898 , AUC = 0.6999998 , bParm = 0.7697799 , dmu/dsigma = 3.06339

mu = 2 , lambda = 5 , nu = 0.3670313 , AUC = 0.6999995 , bParm = 0.7857195 , dmu/dsigma = 3.156897

RsmRocAuc constraint = 0.8

Vary mu and nu only, lambda = 2

mu = 2 , lambda = 2 , nu = 0.5750174 , AUC = 0.8 , bParm = 0.8105541 , dmu/dsigma = 5.945703

mu = 2.214346 , lambda = 2 , nu = 0.4826833 , AUC = 0.7999926 , bParm = 0.7649096 , dmu/dsigma = 5.011078

mu = 2.451665 , lambda = 2 , nu = 0.4128075 , AUC = 0.799998 , bParm = 0.7208098 , dmu/dsigma = 4.438846

mu = 2.714418 , lambda = 2 , nu = 0.3585499 , AUC = 0.7999999 , bParm = 0.6784979 , dmu/dsigma = 4.073575

mu = 3.00533 , lambda = 2 , nu = 0.3154119 , AUC = 0.8000152 , bParm = 0.6382351 , dmu/dsigma = 3.838983

mu = 3.327421 , lambda = 2 , nu = 0.2801812 , AUC = 0.800009 , bParm = 0.6001421 , dmu/dsigma = 3.690938

mu = 3.684031 , lambda = 2 , nu = 0.2506689 , AUC = 0.8 , bParm = 0.5642539 , dmu/dsigma = 3.602765

mu = 4.078861 , lambda = 2 , nu = 0.22533 , AUC = 0.7999993 , bParm = 0.5304907 , dmu/dsigma = 3.556586

mu = 4.516005 , lambda = 2 , nu = 0.2030971 , AUC = 0.7999966 , bParm = 0.4987007 , dmu/dsigma = 3.539988

mu = 5 , lambda = 2 , nu = 0.1833044 , AUC = 0.7999987 , bParm = 0.468758 , dmu/dsigma = 3.545271

Vary mu and lambda only, nu = 1

mu = 1.354458 , lambda = 1 , nu = 1 , AUC = 0.8 , bParm = 0.9446086 , dmu/dsigma = 17.9523

mu = 1.401586 , lambda = 1.195813 , nu = 1 , AUC = 0.8 , bParm = 0.9411753 , dmu/dsigma = 17.10578

mu = 1.450946 , lambda = 1.429969 , nu = 1 , AUC = 0.8000023 , bParm = 0.9378352 , dmu/dsigma = 16.38231

mu = 1.502253 , lambda = 1.709976 , nu = 1 , AUC = 0.8000023 , bParm = 0.9345551 , dmu/dsigma = 15.74782

mu = 1.555229 , lambda = 2.044812 , nu = 1 , AUC = 0.7999999 , bParm = 0.9312388 , dmu/dsigma = 15.16258

mu = 1.609608 , lambda = 2.445213 , nu = 1 , AUC = 0.7999988 , bParm = 0.9276992 , dmu/dsigma = 14.57858

mu = 1.665097 , lambda = 2.924018 , nu = 1 , AUC = 0.7999947 , bParm = 0.9236312 , dmu/dsigma = 13.93974

mu = 1.721499 , lambda = 3.496579 , nu = 1 , AUC = 0.8 , bParm = 0.918575 , dmu/dsigma = 13.18735

mu = 1.77851 , lambda = 4.181255 , nu = 1 , AUC = 0.8000001 , bParm = 0.9119257 , dmu/dsigma = 12.27557

mu = 1.835951 , lambda = 5 , nu = 1 , AUC = 0.8000001 , bParm = 0.9029454 , dmu/dsigma = 11.19209

Vary lambda and nu only, mu = 2

mu = 2 , lambda = 0.1 , nu = 0.464028 , AUC = 0.7999957 , bParm = 0.9476004 , dmu/dsigma = 34.84872

mu = 2 , lambda = 0.1544452 , nu = 0.4672318 , AUC = 0.7999955 , bParm = 0.9265439 , dmu/dsigma = 23.85654

mu = 2 , lambda = 0.2385332 , nu = 0.4721744 , AUC = 0.7999952 , bParm = 0.9009387 , dmu/dsigma = 16.74355

mu = 2 , lambda = 0.3684031 , nu = 0.4797958 , AUC = 0.7999947 , bParm = 0.8726752 , dmu/dsigma = 12.15241

mu = 2 , lambda = 0.568981 , nu = 0.4915404 , AUC = 0.7999938 , bParm = 0.8452798 , dmu/dsigma = 9.214737

mu = 2 , lambda = 0.8787639 , nu = 0.5096299 , AUC = 0.7999923 , bParm = 0.8232878 , dmu/dsigma = 7.386057

mu = 2 , lambda = 1.357209 , nu = 0.5375296 , AUC = 0.7999984 , bParm = 0.8110032 , dmu/dsigma = 6.344095

mu = 2 , lambda = 2.096144 , nu = 0.5806387 , AUC = 0.8 , bParm = 0.8112136 , dmu/dsigma = 5.92933

mu = 2 , lambda = 3.237394 , nu = 0.6480658 , AUC = 0.8000001 , bParm = 0.8241099 , dmu/dsigma = 6.112281

mu = 2 , lambda = 5 , nu = 0.7572216 , AUC = 0.8 , bParm = 0.8448119 , dmu/dsigma = 6.877419

# Online Appendix 17.H: On the continuity of the slope of the RSM-predicted ROC curve at the end-point

As noted in the Errata.pdf document, the book has an incorrect statement regarding continuity of the RSM-predicted ROC curve at the end-point. The following proof of the continuity is adapted from a document supplied by Mr. Xuetong Zhai.

## Online Appendix 17.H.1: Slope of dashed line

The end point of the continuous part of ROC curve is derived in book Equations 17.1 (for maximum FPF), and book Equations 17.2 and 17.4 (for maximum TPF). The final expressions are reproduced below:





Therefore, the slope of the dashed straight line is:

 .

## Online Appendix 17.H.2: Limiting slope of continuous line

On the continuous section,  and  are defined by (book Equations 17.8 and 17.16)

 .

 .

Using Online Appendix Equation 17.2, expressions involving the error function can be replaced by expressions involving the unit normal CDF. For example,



Likewise,

 .

Therefore,

 ,

and

 .

Taking differentials with respect to , from and it follows that:



Factoring,



Simplifying,

 .

Using the following result:

 ,

it follows that

 .



The following code explains the discrepancy between the visual impression, Figure 1, of a discontinuity and the analytical result that the slope is continuous. Open the file mainSlopeContinuity.R, which is based on code designed by Mr. Xuetong Zhai.

## Online Appendix 17.H.3: Code listing

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

library(RJafroc)

library(Rmpfr)

mu <- 0.5

lambda <- 0.1

nu <- 0.8

lambdaP <- lambda / mu

nuP <- 1 - exp(-mu \* nu)

lesionDistr <- rbind(c(1, 0.2), c(2, 0.8))

for (myNegInf in (-3):(-15)) {

ret <- PlotRsmOperatingCharacteristics(

mu, lambda, nu, type = "ROC",

lesionDistribution = lesionDistr,

myNegInf = myNegInf)

cat("myNegInf = ", myNegInf, "\n")

print(ret$ROCPlot)

i <- 1

while (((ret$ROCPlot$data$TPF[i+1] - ret$ROCPlot$data$TPF[1]) == 0) ||

((ret$ROCPlot$data$FPF[i+1] - ret$ROCPlot$data$FPF[1]) == 0)){

i <- i + 1

}

slopeContinuous <- (ret$ROCPlot$data$TPF[i+1]-ret$ROCPlot$data$TPF[1])/

(ret$ROCPlot$data$FPF[i+1]-ret$ROCPlot$data$FPF[1])

maxFPF <- (1 - exp(-lambdaP))

maxTPF <- 1 - (0.2 \* (1 - nuP)^1 \* exp(-lambdaP) +

0.8 \* (1 - nuP)^2 \* exp(-lambdaP))

slopeDashed <- (1 - maxTPF)/(1 - maxFPF)

cat("slopeContinuous = ", slopeContinuous, ", slopeDashed = ", slopeDashed, "\n")

zeta1 <- mpfr(-50, 2000) # set zeta1 = -50 with 2000 digit precision

zeta2 <- zeta1 + 1e-12 # small increment

delta\_FPF <- (1 - exp(-lambdaP\*pnorm(-zeta1))) - (1 - exp(-lambdaP\*pnorm(-zeta2)))

zeta <- zeta1

TPF1 <- 0.2 \* (1 - (1 - nuP \* pnorm(mu-zeta))^1 \* exp(-lambdaP\*pnorm(-zeta))) +

0.8 \* (1 - (1 - nuP \* pnorm(mu-zeta))^2 \* exp(-lambdaP\*pnorm(-zeta)))

zeta <- zeta2

TPF2 <- 0.2 \* (1 - (1 - nuP \* pnorm(mu-zeta))^1 \* exp(-lambdaP\*pnorm(-zeta))) +

0.8 \* (1 - (1 - nuP \* pnorm(mu-zeta))^2 \* exp(-lambdaP\*pnorm(-zeta)))

delta\_TPF <- TPF1 - TPF2

slopeNumeric <- delta\_TPF/delta\_FPF

cat("slopeNumeric = ", as.numeric(slopeNumeric), "\n\n")

}

PlotRsmOperatingCharacteristics calculates FPF and TPF at values of zeta ranging from myNegInf to mu + 3 in steps of 0.01. The default value of myNegInf is -3, which captures almost all of the highest rating distribution on all cases, and should get one quite close to the theoretical end-point. The first iteration of the for-loop, beginning at line 12, uses this value for myNegInf, for which the spacing of the points is such that the while-loop beginning at line 22 is not executed, so i = 1. Line 27 calculates the limiting slope of the continuous section from the position of the uppermost two points, corresponding to i = 1 and i = 2. Source the code to get the output shown below. The output below shows for myNegInf = -3 this slope, slopeContinuous, is 0.9078. The slope of the dashed line is calculated at line 29 – 32. The slope of the dashed line is 0.4935272, which is smaller than the limiting slope of the continuous section, 0.9078, consistent with the visual impression, Figure 1. The reason is that the slope of the continuous section is being estimated from points not sufficiently close to the end-point. Line 35 uses the function mpfr(-50, 2000), for multiple precision numbers, to create zeta1 = -50 with 2000 digit precision. The next line increments this number by 10-12 to create zeta2. Lines 37 – 45 calculates the slope between these two values, and the result is 0.4935272, identical to that of the dashed line. Examination of the output for all values of myNegInf shows that in every case the numerical slope, slopeNumeric, evaluated using high precision calculations to minimize rounding errors, is identical to the slope of the dashed line, slopeDashed. Moreover, slopeContinuous evaluated using the two highest points on the continuous section is variable, settling down to 0.5 when myNegInf is smaller than about -9. [Lines 21 – 25 ensure that a zero divided by zero situations do not occur for very small values of myNegInf.]

## Online Appendix 17.H.4: Code Output

> source(...)

myNegInf = -3

slopeContinuous = 0.9077505 , slopeDashed = 0.4935272

slopeNumeric = 0.4935272

myNegInf = -4

slopeContinuous = 0.7446842 , slopeDashed = 0.4935272

slopeNumeric = 0.4935272

myNegInf = -5

slopeContinuous = 0.6458611 , slopeDashed = 0.4935272

slopeNumeric = 0.4935272

myNegInf = -6

slopeContinuous = 0.5859194 , slopeDashed = 0.4935272

slopeNumeric = 0.4935272

myNegInf = -7

slopeContinuous = 0.5539568 , slopeDashed = 0.4935272

slopeNumeric = 0.4935272

myNegInf = -8

slopeContinuous = 1 , slopeDashed = 0.4935272

slopeNumeric = 0.4935272

myNegInf = -9

slopeContinuous = 0.5 , slopeDashed = 0.4935272

slopeNumeric = 0.4935272

myNegInf = -10

slopeContinuous = 0.5 , slopeDashed = 0.4935272

slopeNumeric = 0.4935272

myNegInf = -11

slopeContinuous = 0.5 , slopeDashed = 0.4935272

slopeNumeric = 0.4935272

myNegInf = -12

slopeContinuous = 0.5 , slopeDashed = 0.4935272

slopeNumeric = 0.4935272

myNegInf = -13

slopeContinuous = 0.5 , slopeDashed = 0.4935272

slopeNumeric = 0.4935272

myNegInf = -14

slopeContinuous = 0.5 , slopeDashed = 0.4935272

slopeNumeric = 0.4935272

myNegInf = -15

slopeContinuous = 0.5 , slopeDashed = 0.4935272

slopeNumeric = 0.4935272

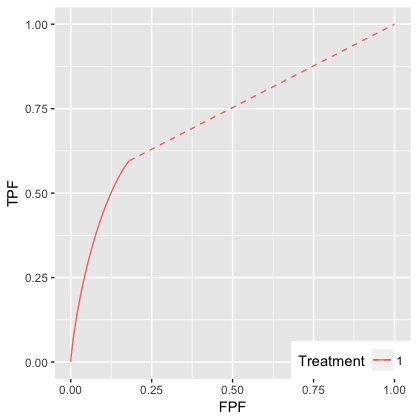


Figure : ROC plot produced for the code listed above. The apparent discontinuity in slope at the end-point is in fact not true, see Online Appendix 17.H.2.

# References

1. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. *Numerical Recipes: The Art of Scientific Computing.* 3 ed. Cambridge: Cambridge University Press; 2007.

2. Chakraborty DP. ROC Curves predicted by a model of visual search. *Phys Med Biol.* 2006;51:3463–3482.

3. Hanley JA. The Robustness of the "Binormal" Assumptions Used in Fitting ROC Curves. *Med Decis Making.* 1988;8(3):197-203.

4. Eng J. ROC analysis: web-based calculator for ROC curves, <http://www.jrocfit.org>. 2006.

1. Maple displays mu, nu, lambda as Greek characters, but will not display primed quantities, at least using my basic knowledge of this tool. [↑](#footnote-ref-1)
2. Maple is not simplifying it further since it has not been told, for example, that  so it need not worry about dividing by zero. Appropriate assume statements could work, but they make the output appear more complicated. [↑](#footnote-ref-2)