Chapter 17: Predictions of the RSM, Online Appendices

Table of Contents

Online Appendix 17.A: The error function

The error function is defined by[1]

$$erf\left(x\right)=\frac{2}{\sqrt{\pi}}\int_{0}^{x}e^{-t^2}\,dt \qquad . \quad \textbf{(17.99.1)}$$

From the definition of the unit normal CDF function $\Phi\left(z\right)\equiv\Phi\left(z;0\right)$,

$$\Phi\left(\sqrt{2}x\right)=\frac{1}{\sqrt{2\pi}}\int_{t=-\infty}^{\sqrt{2}x}e^{-t^2/2}\,dt=\frac{1}{2}+\frac{1}{\sqrt{2\pi}}\int_{t=0}^{\sqrt{2}x}e^{-t^2/2}\,dt$$

$$2\Phi\left(\sqrt{2}x\right)-1=\frac{2}{\sqrt{2\pi}}\int_{t=0}^{\sqrt{2}x}e^{-t^2/2}\,dt$$

$$q=t/\sqrt{2};q^2=t^2/2;UL=x$$

$$2\Phi\left(\sqrt{2}x\right)-1=\frac{2}{\sqrt{2\pi}}\int_{t=0}^{x}e^{-q^2}\,dq\sqrt{2}=\frac{2}{\sqrt{\pi}}\int_{t=0}^{x}e^{-q^2}\,dq=erf\left(x\right)$$

The final result is:

$$erf\left(x\right)=2\Phi\left(\sqrt{2}x\right)-1 \qquad . \quad \textbf{(17.99.2)}$$

■

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.B.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 normal distribution probability density function, the normal cumulative distribution function, the Poisson distribution probability mass function and the binomial distribution probability mass function, respectively. The next line implements Eqn. **Error! Reference source not found.**, the contribution to *TPF* of diseased cases with *n* latent NLs and *l* latent LLs. The next line implements the summation over *n*:

$$TPF(\zeta;\mu,\nu',n)=\sum_{l=0}^{L} f_{bin}\left(l\mid L,\nu'\right)TPF(\zeta;\mu,n,l) \qquad . \quad \textbf{(17.99.3)}$$

The last line implements the summation over n_1:

$$TPF(\zeta;\mu,\lambda',\nu')=\sum_{n=0}^{\infty} f_{Poi}\left(n;\lambda'\right)TPF(\zeta;\mu,\nu',n) \qquad . \quad \textbf{(17.99.4)}$$

The final line displays the result:

$$-\left(\frac{\nu'\operatorname{erf}\left(-\frac{1}{2}\sqrt{2}\left(\zeta-\mu\right)\right)+\nu'-2}{2\nu'-2}\right)^{L}\left(1-\nu'\right)^{L}\mathrm{e}^{\frac{1}{2}\lambda'\left(-1+\operatorname{erf}\left(\frac{1}{2}\sqrt{2}\zeta\right)\right)}+\left(-\frac{1}{-1+\nu'}\right)^{L}\left(1-\nu'\right)^{L}$$

This looks complicated, and probably someone with greater experience with Maple can write the appropriate code to simplify it[1]. I will do it the old fashioned way. The last term is easily seen to be unity. So one is left with:

$$1-\left(\frac{\nu'\operatorname{erf}\left(-\frac{1}{2}\sqrt{2}\left(\zeta-\mu\right)\right)+\nu'-2}{2\nu'-2}\right)^{L}\left(1-\nu'\right)^{N}\mathrm{e}^{\frac{1}{2}\lambda'\left(-1+\operatorname{erf}\left(\frac{1}{2}\sqrt{2}\zeta\right)\right)}$$

Factoring out -2 from the denominator of the large bracketed term, one gets:

$$1-\left(\frac{\nu'\operatorname{erf}\left(-\frac{1}{2}\sqrt{2}\left(\zeta-\mu\right)\right)+\nu'-2}{\left(-2\right)\left(1-\nu'\right)}\right)^{L}\left(1-\nu'\right)^{N}\mathrm{e}^{\frac{1}{2}\lambda'\left(-1+\operatorname{erf}\left(\frac{1}{2}\sqrt{2}\zeta\right)\right)}$$

[1] Maple is not simplifying it further since it has not been told, for example, that $0<\nu'<1$ so it need not worry about dividing by zero. Appropriate **`assume`** statements could work, but they make the output cumbersome.

It is seen that the $\left(1-\nu'\right)^L$ term cancels out leaving:

$$1-\left(\frac{\nu'\operatorname{erf}\left(-\frac{1}{2}\sqrt{2}\left(\zeta-\mu\right)\right)+\nu'-2}{(-2)}\right)^L \mathrm{e}^{\frac{1}{2}\lambda'\left(-1+\operatorname{erf}\left(\frac{1}{2}\sqrt{2}\zeta\right)\right)}$$

Simplifying this leads to:

$$1-\left(-\frac{\nu'}{2}\operatorname{erf}\left(-\frac{1}{2}\sqrt{2}\left(\zeta-\mu\right)\right)-\frac{\nu'}{2}+1\right)^L \mathrm{e}^{\frac{1}{2}\lambda'\left(-1+\operatorname{erf}\left(\frac{1}{2}\sqrt{2}\zeta\right)\right)} \quad . \quad \mathbf{(17.99.5)}$$

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

$$1-\left(\frac{\nu'}{2}\operatorname{erf}\left(\frac{1}{2}\sqrt{2}\left(\zeta-\mu\right)\right)-\frac{\nu'}{2}+1\right)^L \mathrm{e}^{\frac{1}{2}\lambda'\left(-1+\operatorname{erf}\left(\frac{1}{2}\sqrt{2}\zeta\right)\right)} \quad . \quad \mathbf{(17.99.6)}$$

This is identical to Eqn. 9 in Ref. [2], reproduced here for convenience of comparison (with obvious replacements $\lambda \to \lambda'; \nu \to \nu'$ corresponding to the new re-parameterized notation):

$$TPF(\zeta \mid \mu, \lambda', \nu', L) = 1-\left(1-\frac{\nu'}{2}+\frac{\nu'}{2}erf\left(\frac{\zeta-\mu}{\sqrt{2}}\right)\right)^L e^{\left(-\frac{\lambda'}{2}+\frac{\lambda'}{2}erf\left(\frac{\zeta}{\sqrt{2}}\right)\right)} \quad . \quad \mathbf{(17.99.7)}$$

■

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

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

$$A=\frac{\left(-\operatorname{erf}\left(1/2\left(-\zeta+\mu\right)\sqrt{2}\right)\mathrm{e}^{-1/2\zeta^2}\lambda'\nu'+2L\nu\,\mathrm{e}^{-1/2\left(-\zeta+\mu\right)^2}-\mathrm{e}^{-1/2\zeta^2}\lambda'\nu'+2\lambda'\mathrm{e}^{-1/2\zeta^2}\right)\left(1/2\nu'\operatorname{erf}\left(1/2\left(-\zeta+\mu\right)\sqrt{2}\right)+\nu'/2-1\right)^L \mathrm{e}^{1/2\lambda'\left(-1+\operatorname{erf}\left(1/2\sqrt{2}\zeta\right)\right)}}{\sqrt{2\pi}\left(\nu'\operatorname{erf}\left(1/2\left(-\zeta+\mu\right)\sqrt{2}\right)+\nu'-2\right)} \quad . \quad \mathbf{(17.99.8)}$$

$$B = \begin{pmatrix} 2\,L\nu' \mathrm{e}^{-1/2\left(-\zeta+\mu\right)^2} - \mathrm{e}^{-1/2\zeta^2}\lambda'\nu' + 2\,\lambda'\mathrm{e}^{-1/2\zeta^2} \\ -\mathrm{erf}\left(\dfrac{\left(-\zeta+\mu\right)}{\sqrt{2}}\right)\mathrm{e}^{-1/2\zeta^2}\lambda'\nu' \end{pmatrix} \begin{pmatrix} 1/2\,\nu'\mathrm{erf}\left(1/2\left(-\zeta+\mu\right)\sqrt{2}\right) \\ +\nu'/2-1 \end{pmatrix}^{L} \quad . \quad \textbf{(17.99.9)}$$

It can be shown that:

$$\mathrm{pdf}_D\left(\zeta\right) = AB\mathrm{e}^{1/2\,\lambda'\left(-1+\mathrm{erf}\left(1/2\sqrt{2}\zeta\right)\right)} \quad . \quad \textbf{(17.99.10)}$$

■

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

Online Appendix 17.D.1: Code listing

```
# mainRsmPlots.R
rm(list = ls())
library(RJafroc)

K2 <- 700;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)

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 <- RsmOperatingCharacteristics(mu, lambda, nu, type = "ALL", lesionDistribution =
lesionDist, legendPosition  = "none")
  print(ret1$PDFPlot) # pdf plot
  #print(ret1$ROCPlot) # ROC plot
  #print(ret1$AFROCPlot)
  next
  }
}
```

The option `lesionDist`, supplied to `RsmOperatingCharacteristics()` at line 14, `lesionDistribution = lesionDist`, is the normalized histogram of the lesion distribution, 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. However, the reader is encouraged to try different values for $L_{\max}$. Line 14 specifies `type` = `"ALL"` (meaning return *all* types of plots, ROC, FROC, LROC and AFROC; see `RJafroc` documentation) and the result is saved to `ret1`. Line 15 displays the pdf plot and line 16, commented out, would have displayed the ROC plot. To plot a curve one extracts the *correct* plot and `print`s 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.

The code in **mainRsmPdfRoc.R** displays pdfs and ROC plots for the values of μ specified in **muArr**: 0.001, 1, 2, 3, 4 and 5. The remaining *intrinsic* RSM model parameters are defined as: $\lambda = \nu = L_{\max} = 1$. This code was used to generate the ROC curves shown in Fig. 17.1 (A – F) and the *pdfs* shown in Fig. 17.2 (A-F).

Online Appendix 17.D.2: Code listing

```
# mainRsmPdfRoc.R
rm(list = ls())
library(RJafroc)
library(ggplot2)
muArr <- c(0.001,1,2,3,4,5);lambda <- 1;nu <- 1; L <- 1
for (i in 1:length(muArr)) {
  mu <- muArr[i]
  ret1 <- RsmOperatingCharacteristics(
    mu, lambda, nu,
    type = "ALL", lesionDistribution = L, legendPosition  = "none"
  )
  print(ret1$PDFPlot) # pdf plot
  ROCPlot <- ret1$ROCPlot
  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, Treatment =
unique(ret1$ROCPlot$data$Treatment))
    ds <- data.frame(FPF = c(fpfMax, fpfCross), TPF = c(tpfMax, tpfCross), Treatment =
unique(ret1$ROCPlot$data$Treatment))
    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, color =
Treatment), shape = 15, size = 3) +
      geom_line(data = ds, mapping = aes(x = FPF, y = TPF, color = Treatment), linetype = 2) +
      geom_line(data = diagonal, mapping = aes(x = FPF, y = TPF), linetype = 2) +
      geom_text(data = dsText, mapping = aes(x = FPF, y = TPF), label = "d[s]", parse = TRUE, size
= 5)
  }
  print(ROCPlot) # ROC plot
  cat("mu = ", mu,", lambda = ", lambda,
      ", nu = ", nu, ", AUC = ", ret1$aucROC,
      ", fpfMax = ", fpfMax,
      ", tpfMax = ", tpfMax,"\n")
}
```

The *pdf* is displayed at line 12 while the ROC, with the superposed perpendicular line from the end-point to the chance diagonal, representing search performance, is displayed at line 28.

Online Appendix 17.E: Is FROC good?

The code in **mainIsFrocGood.R** was used to generate the plots in Fig. 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 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];lambda <- 1;nu <- 1
  ret1 <- RsmOperatingCharacteristics(
    mu, lambda, nu,
    lesionDistribution = lesionDist, legendPosition  = "none",
    llfRange = c(0,1)
  )
  #print(ret1$FROCPlot) # FROC plot
  #print(ret1$AFROCPlot) # AFROC plot
  #print(ret1$ROCPlot) # ROC plot
  cat("mu = ", mu,", lambda = ", lambda,
      ", nu = ", nu, ", aucFroc = ", ret1$aucFROC,
      ", aucRoc = ", ret1$aucROC,", aucAfroc = ", ret1$aucAFROC,"\n")
  aucRoc[i] <- ret1$aucROC
  aucAfroc[i] <- ret1$aucAFROC
  aucFroc[i] <- ret1$aucFROC
  next
}
cat("approx slope AFROC vs ROC =", (aucAfroc[9]-aucAfroc[1])/(aucRoc[9]-aucRoc[1]),"\n")
#plot curves
aucRocFroc <- data.frame(aucRoc = aucRoc, aucFroc = aucFroc)
plotRocFroc <- ggplot(data = aucRocFroc, mapping = aes(x = aucRoc, y = aucFroc)) + geom_point(size
= 2) + xlab("ROC-AUC") + ylab("FROC-AUC")
print(plotRocFroc)
aucRocAfroc <- data.frame(aucRoc = aucRoc, aucAfroc = aucAfroc)
plotRocAfroc <- ggplot(data = aucRocAfroc, mapping = aes(x = aucRoc, y = aucAfroc)) +
geom_point(size = 2) + xlab("ROC-AUC") + ylab("AFROC-AUC")
print(plotRocAfroc)

## PART 2
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)) {
  mu <- 2;lambda <- lambdaArr[i];nu <- 1
  ret1 <- RsmOperatingCharacteristics(
    mu, lambda, nu,
    lesionDistribution = lesionDist, legendPosition  = "none",
    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
  next
}
cat("approx slope AFROC vs ROC =", (aucAfroc[9]-aucAfroc[1])/(aucRoc[9]-aucRoc[1]),"\n")
# plot curves
aucRocFroc <- data.frame(aucRoc = aucRoc, aucFroc = aucFroc)
plotRocFroc <- ggplot(data = aucRocFroc, mapping = aes(x = aucRoc, y = aucFroc)) + geom_point(size
= 2) + xlab("ROC-AUC") + ylab("FROC-AUC")
print(plotRocFroc)
aucRocAfroc <- data.frame(aucRoc = aucRoc, aucAfroc = aucAfroc)
plotRocAfroc <- ggplot(data = aucRocAfroc, mapping = aes(x = aucRoc, y = aucAfroc)) +
geom_point(size = 2) + xlab("ROC-AUC") + ylab("AFROC-AUC")
print(plotRocAfroc)

## PART 3
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)) {
  mu <- 2;lambda <- 1;nu <- nuArr[i]
  ret1 <- RsmOperatingCharacteristics(
    mu, lambda, nu,
    lesionDistribution = lesionDist, legendPosition  = "none",
    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")
# plot curves
aucRocFroc <- data.frame(aucRoc = aucRoc, aucFroc = aucFroc)
plotRocFroc <- ggplot(data = aucRocFroc, mapping = aes(x = aucRoc, y = aucFroc)) + geom_point(size
= 2) + xlab("ROC-AUC") + ylab("FROC-AUC")
print(plotRocFroc)
aucRocAfroc <- data.frame(aucRoc = aucRoc, aucAfroc = aucAfroc)
plotRocAfroc <- ggplot(data = aucRocAfroc, mapping = aes(x = aucRoc, y = aucAfroc)) +
geom_point(size = 2) + xlab("ROC-AUC") + ylab("AFROC-AUC")
print(plotRocAfroc)
```

The code is divided into 3 parts: Part I calculates $AUC_{RSM}^{FROC}\left(\mu,\lambda,\nu\right)$, $AUC_{RSM}^{ROC}\left(\mu,\lambda,\nu,f_L\right)$ and $AUC_{RSM}^{AFROC}\left(\mu,\lambda,\nu\right)$ for varying μ, values specified at line 17, with $\lambda=\nu=1$; Part II calculates the same AUCs for varying λ, values specified at line 48, with $\mu=2;\nu=1$; and Part III calculates them for varying ν, values specified at line 76, with $\mu=2;\lambda=1$. The in-line defined function **`logseq()`** creates an array with logarithmically spaced values.

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

The following code generates ROC data for 500 non-diseased and 700 diseased cases, for 4 sets of RSM parameter values, the same values used to generate the *pdfs* shown in shown in Fig. 17.2 (A - D). The 5-bin ratings data is fitted by **`RocfitR()`** and pairs of RSM predicted and binormal model fitted ROC curves are displayed for easy comparison. The resulting ROC plots are shown Fig. 17.1 (A - D).

Online Appendix 17.F.1: Code Listing

```
# mainRsmVsRocfitR.R
rm(list = ls())
library(RJafroc)
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){
      RowString <- toString(c(seed,i,j))
      mu <- muArr[i];nu <- nuArr[j]

      frocDataRaw  <- FROCSimulator (mu, lambda, nu, K1, K2, Lk2, zeta1 = zeta1)
      rocDataRaw <- FROC2HrROC(frocDataRaw)

      rocDataBinned <- BinDataset(rocDataRaw,nBins)
      if (length(unique(rocData$LL[1,1,,1])) != nBins) stop("too few bins")
      rocDataBinned <- rocDataBinned

      rocDataTable <- array(dim = c(2,nBins))
      rocDataTable[1,] <- table(rocDataBinned$NL[1:K1])
      rocDataTable[2,] <- table(rocDataBinned$LL[1:K2])

      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
    }
  }
}
```

Line 20 initializes the **seed** variable (2 or 3). Changing **seed** is equivalent to sampling a fresh dataset. FROC data is simulated at line 31 and converted to ROC data at line 32. 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 34 - 36) may not find 5 bins; notice the check at line 35 for too few bins. Lines 38 – 40 constructs the ROC data table, the analog of Table 4.1. The binormal model fitting function, **RocfitR()**, is called at line 42, followed by a check to see if the algorithm converged, line 43. The binormal model fitting part of the code (lines 34 - 43) should be familiar from Chapter "Binormal Model". Line 45 calls the function **AucsRsm()** which numerically integrates the ROC and AFROC curves, including any applicable straight line extensions. Lines 47-49 displays the combined ROC plots.

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

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 Table 4.1. These are analyzed by the Eng Java program[37], see §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 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 50 - 58.

```
# mainRsmVsEng.R
rm(list = ls())
library(RJafroc)
library(ggplot2)
library(binom)
source("AucsRsm.R")
source("PlotBMErrBar.R")

K1 <- 500;K2 <- 700;nBins <- 5;
Row <- 1
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;set.seed(seed);Lmax <- 2;mu <- 3.0;lambda <-  1;nu <- 1;zeta1 <- -1},# Row 
8
       "9" = {seed <- 2;set.seed(seed);Lmax <- 2;mu <- 3.0;lambda <- 0.1;nu <- 1;zeta1 <- -1}# Row 
9
)
cat("K1 = ", K1, ", K2 = ", K2, ", zeta1 = ", zeta1, ", seed = ", seed,
    ", Lmax = ", Lmax, ", mu = ", mu, ", lambda = ", lambda, ", nu = ", 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  <- FROCSimulator (mu, lambda, nu, K1, K2, Lk2, zeta1 = zeta1)
rocDataRaw <- FROC2HrROC(frocDataRaw)

rocDataBinned <- BinDataset(rocDataRaw,nBins)
if (length(unique(rocData$LL[1,1,,1])) != nBins) stop("too few bins")
rocDataBinned <- rocDataBinned

rocDataTable <- array(dim = c(2,nBins))
rocDataTable[1,] <- table(rocDataBinned$NL[1:K1])
rocDataTable[2,] <- table(rocDataBinned$LL[1:K2])

aucs <- AucsRsm(mu = mu, lambda = lambda, nu = nu, lesionDist = lesionDist)
cat("RSM-ROC-AUC = ", aucs$aucROC, "\n")
print(rocDataTable)
# 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 in book

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 <- 1.002;b <- 0.861}, # Row 1
       "2" = {a <- 1.497;b <- 0.752}, # Row 2
       "3" = {a <- 1.928;b <- 0.736}, # Row 3
       "4" = {a <- 1.221;b <- 0.682}, # Row 4
       "5" = {a <- 1.250;b <- 0.786},# Row 5
       "6" = {a <- 1.554;b <- 0.646},# Row 6
       "7" = {a <- 2.057;b <- 0.676},# Row 7
       "8" = {a <- 2.391;b <- 0.405},# Row 8
       "9" = {a <- 2.015;b <- 0.068}# Row 9
)

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

The **`switch()`** statement allows selection of a specific set of parameters depending on the value of **`Row`** specified at line 10. For example, if **`Row <- 1`**, 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 $K_1 = 500, K_2 = 700$ and $\zeta_1 = -1$. 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 1 count in both TP and FP categories (search **`RJafroc`** documentation for **`BinDataset()`**for details).

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 al[3] that the b-parameter decreases with increasing lesion contrast and the ratio $\Delta(mean)/\Delta(\sigma)$is approximately constant for a fixed set of experimental conditions. The results of sourcing this code were used to populate 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 Table 17.1 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 $\mu \& \nu$ for constant AUC for $\lambda = 2$. Part B, lines 36 – 53, varies $\mu \& \lambda$ for constant AUC for $\nu = 1$. Part C, lines 54 – 73, varies $\lambda \& \nu$ for constant AUC for $\mu = 2$. 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 $\Delta\left(mean\right)/\Delta\left(\sigma\right)$. A seed variable is not needed in this function, as random samples are not involved.

17.H: 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. Swets JA, Tanner Jr WP, Birdsall TG. Decision processes in perception. *Psychological review.* 1961;68(5):301.