**Chapter 19 Online Appendices**

# Online Appendix 19.A: Fitting 3 proper ROC models to the same data

The relevant book section is 19.5; the code file is mainRSM.R listed below, which fits CBM and RSM models for each modality-reader combination contained in 14 MRMC datasets. In addition, it reads results of PROPROC fits to the same datasets, which were obtained using OR-DBM-MRMC software run on a Windows 8 virtual machine under OSX (book Chapter 21 has technical details of how this was done). Comprehensive results of the analysis, including paramters and generated ROC plots, are saved to allResults, whose structure is shown in Table 1. Ensure that f is set to 1 at line 22. Source the code to get the output shown in book section 19.5.1.1, corresponding to dataset01, and 10 plots in the Plots window, corresponding to the 2 modalities times 5 readers in this dataset. Each plot consists of RSM (red), CBM (blue) and PROPROC (black) fits to the same ROC dataset. Plots for the 14 datasets (identified at line 16 - 20) are in the online document RSM PROPROC CBM 236 Plots.pdf.

## Online Appendix 19.A.1 Code listing mainRSM.R

# mainRSM.R

# RSM fits vs. PROPROC and CBM fits;

# Windows proproc results must be saved to MRMCRuns

# directory prior to running this

rm(list = ls())

library(RJafroc)

library(ggplot2)

library(bbmle)

library(stats)

library(binom)

options(digits = 3)

reAnalyze <- FALSE;showPlot <- TRUE;saveProprocLrcFile <- FALSE

# included datasets

fileNames <- c("TONY", "VD", "FR",

"FED", "JT", "MAG",

"OPT", "PEN", "NICO",

"RUS", "DOB1", "DOB2",

"DOB3", "FZR")

f <- 1 # selected dataset

fileName <- fileNames[f]

# the datasets already exist as R objects

theData <- get(sprintf("dataset%02d", f))

# RSM ROC fitting needs to know lesionDistribution

lesionDistribution <- UtilLesionDistribution(theData)

rocData <- DfFroc2Roc(theData)

if (saveProprocLrcFile) {

DfSaveDataFile(rocData,

fileName =

paste0(fileName,".lrc"),

format = "MRMC")

}

I <- length(rocData$modalityID);J <- length(rocData$readerID)

K <- dim(rocData$NL)[3];K2 <- dim(rocData$LL)[3];K1 <- K - K2

## retrieve PROPROC parameters

csvFileName <- paste0(fileName, " proproc area pooled.csv")

sysCsvFileName <- system.file(

paste0(

"MRMCRuns/",fileName), csvFileName, package = "RJafroc")

if (!file.exists(sysCsvFileName))

stop("Run Windows PROPROC for this dataset using VMwareFusion")

proprocRet <- read.csv(sysCsvFileName)

c1 <- matrix(data =

proprocRet$c,

nrow = length(unique(proprocRet$T)),

ncol = length(unique(proprocRet$R)), byrow = TRUE)

da <- matrix(data =

proprocRet$d\_a,

nrow = length(unique(proprocRet$T)),

ncol = length(unique(proprocRet$R)), byrow = TRUE)

retFileName <- paste0("allResults", fileName)

sysAnalFileName <- system.file(

"ANALYZED/RSM6", retFileName, package = "RJafroc")

if (fileName %in% c("JT", "NICO", "DOB1", "DOB3")){

binnedRocData <- DfBinDataset(rocData, desiredNumBins = 5, opChType = "ROC")

}else{

binnedRocData <- rocData

}

if (reAnalyze || !file.exists(sysAnalFileName)){

allResults <- list()

AllResIndx <- 0

for (i in 1:I){

for (j in 1:J){

AllResIndx <- AllResIndx + 1

cat("f, i, j:", f, i, j, "\n")

# fit to CBM

retCbm <- FitCbmRoc(

binnedRocData, trt = i, rdr = j)

# fit to RSM, need lesionDistribution matrix

retRsm <- FitRsmRoc(

binnedRocData, trt = i, rdr = j, lesionDistribution)

if (showPlot) {

x <- allResults[[AllResIndx]]

lesionDistribution <- x$lesionDistribution

empOp <- UtilBinCountsOpPts(binnedRocData, trt = i, rdr = j)

fpf <- empOp$fpf; tpf <- empOp$tpf

compPlot <- gpfPlotRsmPropCbm(

which(fileNames == fileName),

x$retRsm$mu, x$retRsm$lambdaP, x$retRsm$nuP,

lesionDistribution, c1[i, j], da[i, j],

x$retCbm$mu, x$retCbm$alpha,

fpf, tpf, i, j, K1, K2, c(1, length(fpf)))

print(compPlot)

}

}

}

# safety comments

# sysSavFileName <-

# paste0("/Users/Dev/rjafroc/inst/ANALYZED/RSM6/", retFileName)

# save(allResults, file = sysSavFileName)

}else{

load(sysAnalFileName)

AllResIndx <- 0

# cat(fileName, "i, j, mu, lambdaP, nuP, c, da, alpha,")

# cat("muCbm, AUC-RSM, AUC-PROPROC, AUC-CBM, chisq, p-value,df\n")

for (i in 1:I){

for (j in 1:J){

AllResIndx <- AllResIndx + 1

x <- allResults[[AllResIndx]]

if (showPlot) {

empOp <- UtilBinCountsOpPts(binnedRocData, trt = i, rdr = j)

fpf <- empOp$fpf; tpf <- empOp$tpf

compPlot <- gpfPlotRsmPropCbm(

which(fileNames == fileName),

x$retRsm$mu, x$retRsm$lambdaP, x$retRsm$nuP,

lesionDistribution = lesionDistribution, c1[i, j], da[i, j],

x$retCbm$mu, x$retCbm$alpha,

fpf, tpf, i, j, K1, K2, c(1, length(fpf)))

compPlot <- compPlot +

theme(

axis.text=element\_text(size=10),

axis.title=element\_text(size=28,face="bold"))

print(compPlot)

}

# follows same format as RSM6 Vs. Others.xlsx

cat(fileName, i, j, x$retRsm$mu, x$retRsm$lambdaP, x$retRsm$nuP,

c1[i,j], da[i,j],

x$retCbm$alpha, x$retCbm$mu,

x$retRsm$AUC, x$aucProp, x$retCbm$AUC,

x$retRsm$ChisqrFitStats[[1]], x$retRsm$ChisqrFitStats[[2]],

x$retRsm$ChisqrFitStats[[3]],"\n")

}

}

}

## 

## Online Appendix 19.A.2 Organization of allResults

Table 1: Results of RSM, CBM and PROPROC analyses contained in the object allResults with the following orgainzation for the first dataset, i.e., dataset 1, modality 1 and *reader 1*. As this is the first dataset, one uses allResults[[1]] to get its contents (double brackets are used to index dataframes). The results for dataset 1, modality 1 and *reader 2* are accessed using allResults[[2]], etc., until the last dataset, corresponding to dataset 14, modality 2 and reader 4, accessed using allResults[[236]]. For example, allResults[[1]]$retRsm$mu (no spaces allowed and R is case sensitive) is the RSM  parameter for the first dataset. As another example, allResults[[1]]$retRsm$ChisqrFitStats$pValis the p-value of the chisquare goodness of fit for this indiviual dataset. The lesion distribution matrix, i.e., the number of diseased cases with 1 lesion, the number with 2 lesions, etc., is only needed for RSM fitting (as the likelihood function depends on it). As far as CBM and PROPROC are concerned, the concept of lesion is irrelevant, as a case is either non-diseased or diseased.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| allResults[[1]] | $retRsm | $mu |  | RSM |
| $lambdaP |  | RSM |
| $nuP |  | RSM |
| $zetas |  | Thresholds |
| $AUC |  | RSM-ROC-AUC |
| $StdAUC |  | Standard deviation of AUC |
| $NLLIni |  | Initial value of negative LL |
| $NLLFin |  | Final value of negative LL |
| $ChisqrFitStats | $chisq | Chisqure goodness of fit statistic |
| $pVal | P-value |
| $df | Degrees of freedom |
| $covMat |  | Covariance matrix |
| $fittedPlot |  | Fitted ROC plot with op. pts. |
| $retCbm | $mu |  | CBM |
| $alpha |  | CBM |
| $zetas |  | Thresholds |
| $AUC |  | CBM-ROC-AUC |
| $StdAUC |  | Standard deviation of AUC |
| $NLLIni |  | Initial value of negative LL |
| $NLLFin |  | Final value of negative LL |
| $ChisqrFitStats | $chisq | Chisqure goodness of fit statistic |
| $pVal | P-value |
| $df | Degrees of freedom |
| $covMat |  | Covariance matrix |
| $fittedPlot |  | Fitted ROC plot with op. pts. |
| $lesionDistribution |  | | Lesion distribution matrix |
| $c1 | Calculated by Windows OR-DBM-MRMC software | | PROPROC *c*-parameter |
| $da | PROPROC  -parameter |
| $aucProp | PROPROC-ROC-AUC |
| $I |  | | # Modalities |
| $J |  | | # Readers |
| $K1 |  | | # Non-diseased cases |
| $K2 |  | | # Diseased cases |

To understand this code, notice that f = 1 at line 22, corresponding to the first dataset (fileNames = "TONY"), insert a break point at line 25 and click Source. The Environment panel is shown in Figure 1.

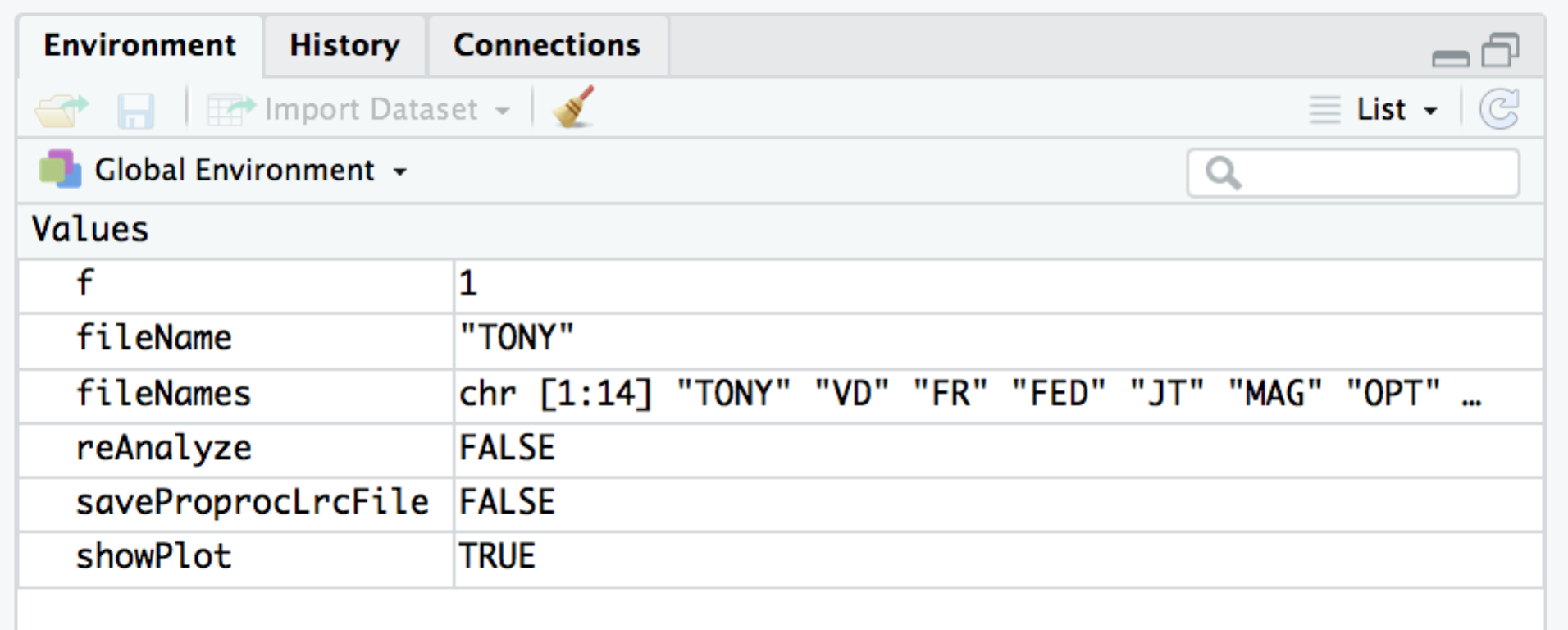


Figure 1: Environment panel with code pointer at line 25.

Three flag variables are seen: reAnalyze = FALSE, saveProprocLrcFile = FALSE, and showPlot = TRUE. Ensure that the values are set to these values (the explanation that follows changes one of them and sourcing the code could then lead to an error condition being displayed).

* The first flag, reAnalyze, if set to TRUE, reanalyzes all datasets; this is intended for users seeking greater understanding of the methods but which runs the risk of overwriting the existing preanalyzed results files. As a further precaution, lines 96 – 98 are commented to prevent accidental overwriting of the results files containd in "~/rjafroc/inst/ANALYZED/RSM6/". The current (at the time of publication of the book) contents of this directory are listed in book Figure 19.4.
* The second flag, saveProprocLrcFile, if set to TRUE, creates .lrc files in the appropriate format for analysis using OR DBM MRMC software, running under Windows 8. This option was selected to analyze the .lrc files using PROPROC, the corresponding results being copied to "~/rjafroc/inst/MRMCRuns/", shown in Figure 2. This option should be used to analyze a new dataset, and the results are copied to this directory. For example, a new dataset could be created by appending "newData" to line 20, which would correspond to f = 15. The folder names in Figure 2 are determined by the contents of fileNames.
* The third flag, showPlot, if set to TRUE, displays plots like those in RSM PROPROC CBM 236 Plots.pdf, in the Plots window.

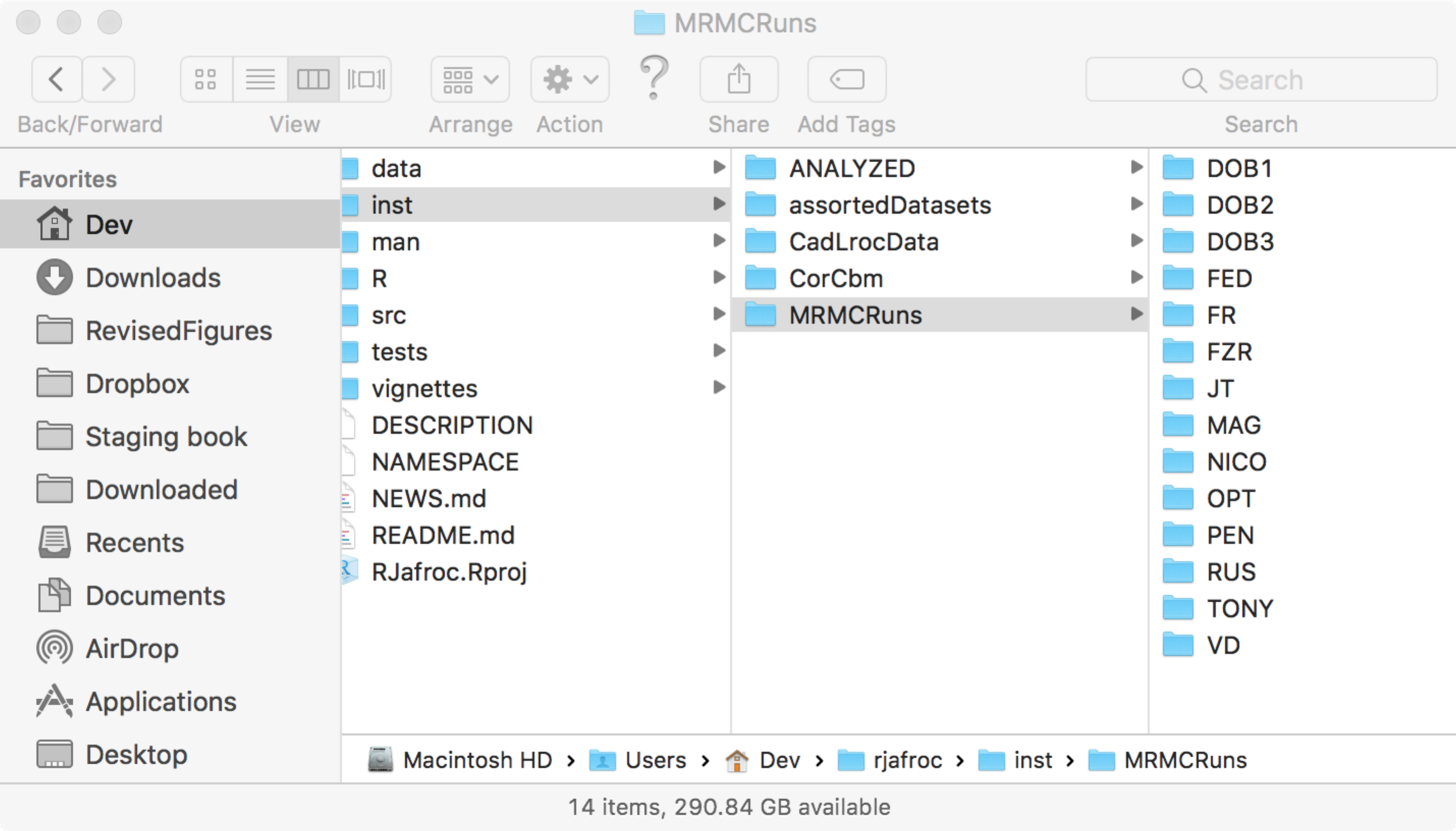


Figure 2: Location of pre-analyzed results obtained by running PROPROC software on the datasets.

Returning to the explanation of the code, the code pointer is at line 25. Clicking Next causes execution of this line, which uses the get() function to find the system file corresponding to f = 1. A new object, theData, appears in the Environment panel. Examination of this object reveals an FROC dataset consisting of two modalities, five readers, 96 non-diseased and 89 diseased cases. The code pointed is at line 27, execution of which (click Next) initializes lesionDistribution.

Browse[2]> lesionDistribution

[,1] [,2]

[1,] 1 83

[2,] 2 6

Its value shows that 83 cases contained one lesion and 6 cases contained 2 lesions. Click Next to execute line 29, which converts the dataset to an ROC dataset, rocData. This object appears in the Environment panel, and the reader should confirm that it is indeed an ROC dataset (the dataType field is "ROC"). Use statements like those shown below to confirm that the highest rating is indeed being used.

Browse[2]> theData$NL[1,1,1:10,]

[,1] [,2] [,3]

[1,] 3.0 3 -Inf

[2,] 3.0 -Inf -Inf

[3,] -Inf -Inf -Inf

[4,] -Inf -Inf -Inf

[5,] 3.0 -Inf -Inf

[6,] 3.5 -Inf -Inf

[7,] 3.0 -Inf -Inf

[8,] -Inf -Inf -Inf

[9,] -Inf -Inf -Inf

[10,] 5.0 -Inf -Inf

Browse[2]> rocData$NL[1,1,1:10,]

[1] 3.0 3.0 2.0 2.0 3.0 3.5 3.0 2.0 2.0 5.0

The alert reader will wonder why case 3, with no marks, was assigned rating 2. This is because this dataset contains clinical mammography ratings. BIRADS rating 1, corresponding to definitely non-diseased, does not generate any marks, and therefore does not appear in the dataset. Likewise, BIRADS rating 2, corresponding to definitely benign, also does not appear in the dataset. The lowest rating in the FROC dataset is actually 3, see code snippet below.

Browse[2]> min(theData$NL[theData$NL != -Inf],theData$LL[theData$LL != -Inf])

[1] 3

In future analysis of such data, the author recommends including BIRADS rating 2 marks, appropriately classified as NLs (which is expected most of the time) or as LLs (the occasional BIRADS 2 marks that happen to find true malignant lesions).

Returning to the explanation, the code pointer is at line 31. Since saveProprocLrcFile is FALSE, lines 31 – 36 are not executed. Click Next enough times to initialize I, J, K1, K2 and K.

The code pointer is at line 41. This initializes the name of the file containing the results of PROPROC analysis. Click Next enough times to advance the code pointer to line 48. Notice the check at line 45 – 46 for existence of the preanalzed results file – if the corresponding .csv file does not exist, program execution will stop here.

proprocRet contains the contents of the .csv file that was generated by running PROPROC. Its contents are listed below.

Browse[2]> proprocRet

T R returnCode area numCAT adjPMean c d\_a X

1 1 1 0 0.801 6 0.801 -0.1323 1.197 NA

2 1 2 0 0.895 5 0.895 -0.0870 1.771 NA

3 1 3 0 0.853 6 0.853 -0.1444 1.482 NA

4 1 4 0 0.858 6 0.858 0.0805 1.514 NA

5 1 5 0 0.891 6 0.891 0.2226 1.740 NA

6 2 1 0 0.672 6 0.672 -0.0817 0.628 NA

7 2 2 0 0.754 5 0.754 0.0498 0.974 NA

8 2 3 0 0.793 6 0.793 -0.1326 1.156 NA

9 2 4 0 0.874 6 0.874 0.1182 1.620 NA

10 2 5 0 0.736 6 0.736 0.0781 0.893 NA

The necessary parameters, c and d\_a are listed in the two second-last columns. Click Next twice to extract these parameters into varaibles c1 and da (minor complication: since c() is the concatenation operator, it cannot be used as a variable name). The extracted values are listed below.

Browse[2]> c1

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

[1,] -0.1323 -0.0870 -0.144 0.0805 0.2226

[2,] -0.0817 0.0498 -0.133 0.1182 0.0781

Browse[2]> da

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

[1,] 1.197 1.771 1.48 1.51 1.740

[2,] 0.628 0.974 1.16 1.62 0.893

The code pointer should be at line 57. This and the following line initialize the names of the results file. Click Next twice.

Browse[2]> retFileName

[1] "allResultsTONY"

Browse[2]> sysAnalFileName

[1] "/Library/Frameworks/R.framework/Versions/3.4/Resources/library/RJafroc/ANALYZED/RSM6/allResultsTONY"

The results file is a "system file" that is buried deep in the installation files corresponding to this software, in other words, it is part of RStudio and RJafroc.

The code pointer is at line 61. If the data is already binned, the binning at line 62 is skipped. Click Next twice to advance to code pointer to line 67. A new object binnedRocData appears in the Environment window. Since the dataset in question is already binned (BIRADS ratings) the contents of binnedRocData are identical to rocData.

The if clause at line 67 evaluates to FALSE (because the results file exists and reAnalyze is FALSE).

Browse[2]> (reAnalyze || !file.exists(sysAnalFileName))

[1] FALSE

Clicking Next advances the code to line 100. This loads the system file sysAnalFileName. The object allResults appears in the Environment panel, Figure 3.

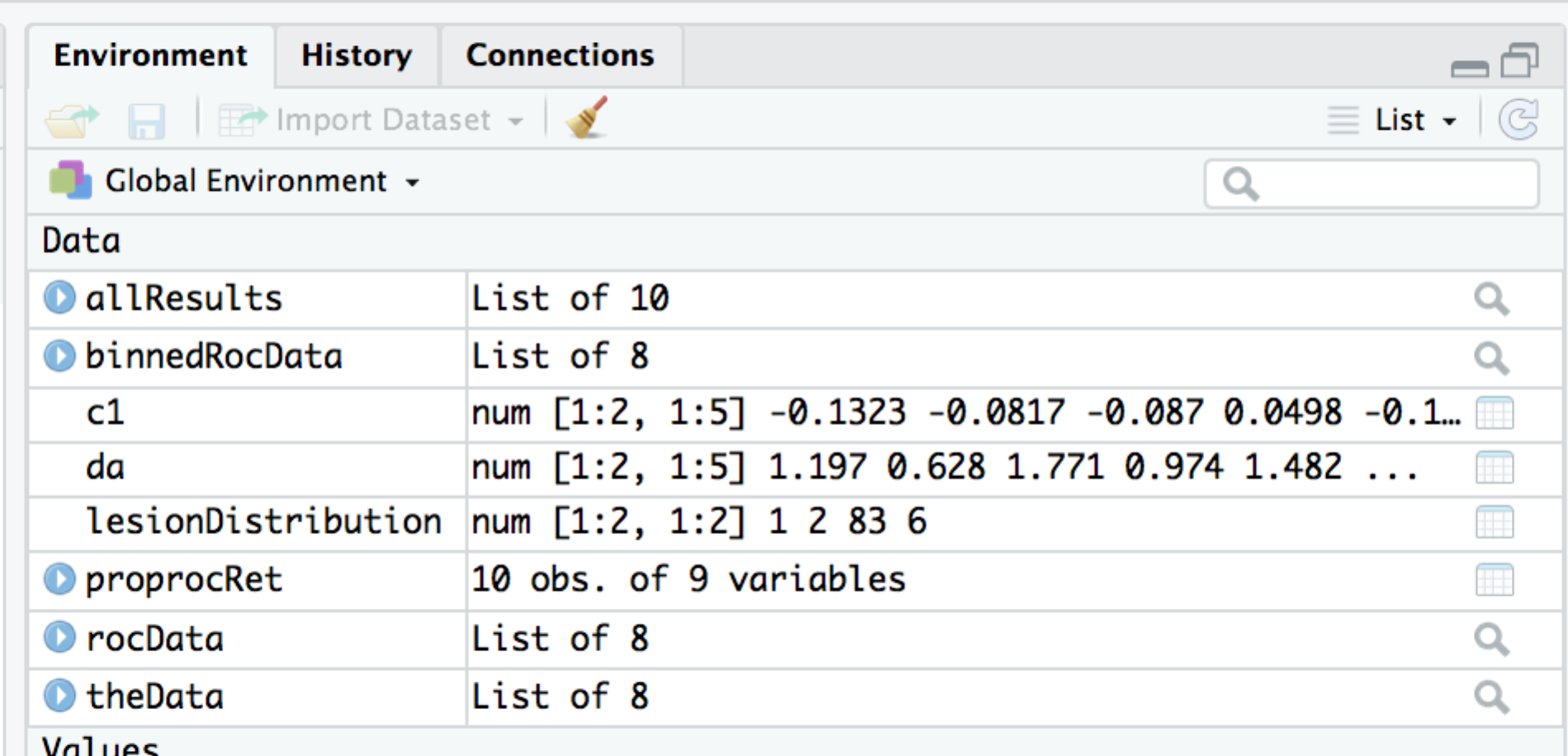


Figure 3: Environment panel with code pointer at line 101.

The allResults object contains the results of analysis for f = 1. Click Next to advance the code pointer to line 104. This line initiates two for-loops, one in i (modality) and the other in j (reader). Click Next enough times to advance the code pointer to line 108. The variable x is shorthand for allResults[[1]]. Click Next to enter the plotting code, then exit the loop by clicking on the "Execute the remainder of the current function or loop" button. The code pointer is now at line 105 and the following plot, Figure 4, appears in the Plots window.

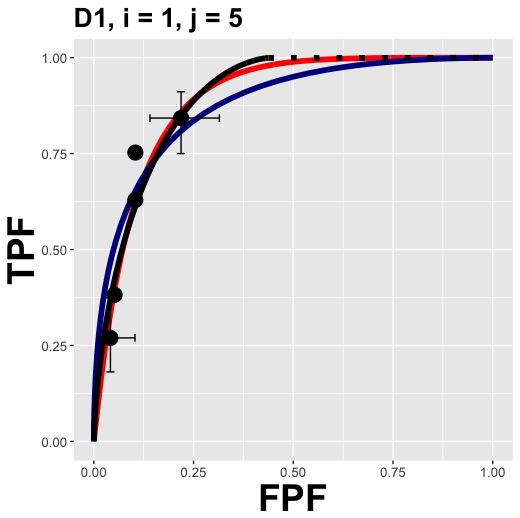


Figure 4: ROC plots for dataset 1, modality 1 and reader 1.

The code repeats for the next reader, etc., and when all readers are plotted, the procedure is repeated for the next modality.

Exit debug mode (click on Stop), remove any breakpoints, and click Source to get all plots for this dataset and the following output.

> source('~/onlinebookk21778/Ch19/software/mainRSM.R')

TONY 1 1 1.78 1.04 0.729 -0.132 1.2 0.775 1.83 0.813 0.801 0.812 1.96 0.161 1

TONY 1 2 1.97 0.841 0.867 -0.087 1.77 0.924 2.04 0.891 0.895 0.893 NA NA NA

TONY 1 3 2.85 19.7 0.939 -0.144 1.48 0.855 1.87 0.853 0.853 0.849 0.305 0.581 1

TONY 1 4 1.22 1.06 0.999 0.0805 1.51 0.999 1.49 0.865 0.858 0.853 6.23 0.0126 1

TONY 1 5 1 0.575 0.999 0.223 1.74 0.999 1.65 0.896 0.891 0.878 4.6 0.032 1

TONY 2 1 1.21 1.06 0.478 -0.0817 0.628 0.598 1.17 0.678 0.672 0.676 2.1 0.147 1

TONY 2 2 1.05 2.35 0.998 0.0498 0.974 0.998 0.951 0.758 0.754 0.749 NA NA NA

TONY 2 3 2.03 2.42 0.729 -0.133 1.16 0.754 1.72 0.793 0.793 0.792 1.36 0.507 2

TONY 2 4 1.03 0.653 0.999 0.118 1.62 0.999 1.51 0.888 0.874 0.857 NA NA NA

TONY 2 5 0.777 1.89 0.998 0.0781 0.893 0.998 0.845 0.741 0.736 0.725 0.912 0.34 1

To understand the fitting section of the code (as opposed to simply accepting the pre-analyzed results), set the reAnalyze flag to TRUE to enter lines 67 – 99 of the code. Set reAnalyze <- TRUE, delete all plots, insert a break point at line 75 and click Source. The code about to be executed fits CBM to the ROC data, FitCbmRoc(). Click Next twice to execute the CBM fit and the RSM fit, FitRsmRoc(). The code pointer is at line 80, which, when executed, shows the plot.

Exit debug mode (Stop button), restore reAnalyze <- FALSE and close the file mainRSM.R. If this is not done, the next time one carelessly sources the code, an error message will appear:

> source(…)

f, i, j: 1 1 1

Error in allResults[[AllResIndx]] : subscript out of bounds

# Online Appendix 19.B: Inter-correlations

This relates to book section 19.5.2, which deals with (inter) correlations between the three methods of estimating AUC and between model parameters with similar meanings yielded by RSM and CBM methods. Following is a listing of mainInterCorrelations.R.

## Online Appendix 19.B.1 Code Listing

# mainInterCorrelations.R

# compares RSM to PROPROC and CBM

# uncomment line 29 to get Fig. 19.6, a,b

# Fig. 19.6c,d

# Fig. 19.7a,b,c,d

rm(list = ls())

library(foreach)

library(RJafroc)

library(doRNG)

library(doParallel)

library(ggplot2)

type <- "pearson";showPlot <- FALSE

fileNames <- c("TONY", "VD", "FR",

"FED", "JT", "MAG",

"OPT", "PEN", "NICO",

"RUS", "DOB1", "DOB2",

"DOB3", "FZR")

options(digits = 6)

clusterParms <- list() # parameters passed to cluster

avgAucPro <- array(dim = length(fileNames))

avgAucCbm <- avgAucPro;

avgAucRsm <- avgAucPro

avgSlopeProRsm <- avgAucPro

avgSlopeCbmRsm <- avgAucPro

avgR2ProRsm <- avgAucPro;avgR2CbmRsm <- avgAucPro

rhoMuRsmMuCbm <- avgAucPro;rhoNupRsmAlphaCbm <- avgAucPro

for (f in 1:length(fileNames)){

#if (f != 7) next

fileName <- fileNames[f]

retFileName <- paste0("allResults", fileName)

sysAnalFileName <- system.file("ANALYZED/RSM6", retFileName, package = "RJafroc")

if (file.exists(sysAnalFileName)){

load(sysAnalFileName)

I <- allResults[[1]]$I

J <- allResults[[1]]$J

aucRsm <- array(dim = c(I, J));aucCbm <- array(dim = c(I, J));aucPro <- array(dim = c(I, J))

muRsm <- array(dim = c(I, J));muCbm <- array(dim = c(I, J))

nupRsm <- array(dim = c(I, J));alphaCbm <- array(dim = c(I, J))

AllResIndx <- 0

for (i in 1:I){

for (j in 1:J){

AllResIndx <- AllResIndx + 1

aucRsm[i, j] <- allResults[[AllResIndx]]$retRsm$AUC

aucPro[i, j] <- allResults[[AllResIndx]]$aucProp

aucCbm[i, j] <- allResults[[AllResIndx]]$retCbm$AUC

muRsm[i, j] <- allResults[[AllResIndx]]$retRsm$mu

nupRsm[i, j] <- allResults[[AllResIndx]]$retRsm$nuP

muCbm[i, j] <- allResults[[AllResIndx]]$retCbm$mu

alphaCbm[i, j] <- allResults[[AllResIndx]]$retCbm$alpha

}

}

# constrained fit thru origin; aucPro vs. aucRsm

df <- data.frame(aucRsm = as.vector(aucRsm), aucPro = as.vector(aucPro))

mProRsm <- lm(aucPro ~ 0 + aucRsm, data = df)

avgSlopeProRsm[f] <- coef(mProRsm)

avgR2ProRsm[f] <- summary(mProRsm)$r.squared

ij <- paste0("D", f, ", I = ", I, ", J = ", J)

p <- ggplot(data = df, aes(x = aucRsm, y = aucPro)) +

geom\_smooth(method = "lm", se = FALSE, color = "black", formula = y ~ 0 + x) +

geom\_point(size = 5) +

labs(title = ij) +

theme(plot.title = element\_text(hjust = 0.5, size = 20, face = "bold")) +

theme(

axis.text=element\_text(size=10),

axis.title=element\_text(size=28,face="bold"))

if (showPlot) print(p)

# constrained fit thru origin; aucCbm vs. aucRsm

df <- data.frame(aucRsm = as.vector(aucRsm), aucCbm = as.vector(aucCbm))

mCbmRsm <- lm(aucCbm ~ 0 + aucRsm, data = df)

avgSlopeCbmRsm[f] <- coef(mCbmRsm)

avgR2CbmRsm[f] <- summary(mCbmRsm)$r.squared

p <- ggplot(data = df, aes(x = aucRsm, y = aucCbm)) +

geom\_smooth(method = "lm", se = FALSE, color = "black", formula = y ~ 0 + x) +

geom\_point(size = 5) +

labs(title = ij) +

theme(plot.title = element\_text(hjust = 0.5, size = 20, face = "bold")) +

theme(

axis.text=element\_text(size=10),

axis.title=element\_text(size=28,face="bold"))

if (showPlot) print(p)

df <- data.frame(muCbm = as.vector(muCbm), muRsm = as.vector(muRsm))

ij <- paste0("D", f, ", I = ", I, ", J = ", J)

p <- ggplot(data = df, aes(x = muRsm, y = muCbm)) +

geom\_smooth(method = "lm", se = FALSE, color = "black", formula = y ~ 0 + x, size = 2) +

geom\_point(size = 5) +

labs(title = ij) +

theme(plot.title = element\_text(hjust = 0.5, size = 20, face = "bold")) +

scale\_color\_manual(values = "black") +

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

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

if (showPlot) print(p)

df <- data.frame(alphaCbm = as.vector(alphaCbm), nupRsm = as.vector(nupRsm))

ij <- paste0("D", f, ", I = ", I, ", J = ", J)

p <- ggplot(data = df, aes(x = nupRsm, y = alphaCbm)) +

geom\_smooth(method = "lm", se = FALSE, color = "black", formula = y ~ 0 + x, size = 2) +

geom\_point(size = 5) +

labs(title = ij) +

theme(plot.title = element\_text(hjust = 0.5, size = 20, face = "bold")) +

scale\_color\_manual(values = "black") +

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

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

if (showPlot) print(p)

avgAucPro[f] <- mean(aucPro)

avgAucCbm[f] <- mean(aucCbm)

avgAucRsm[f] <- mean(aucRsm)

rhoMuRsmMuCbm[f] <- cor(as.vector(muRsm), as.vector(muCbm),method = "pearson")

rhoNupRsmAlphaCbm[f]<- cor(as.vector(nupRsm), as.vector(alphaCbm),method = "pearson")

cat(avgAucPro[f],avgAucCbm[f],avgAucRsm[f],

avgSlopeProRsm[f],avgR2ProRsm[f],

avgSlopeCbmRsm[f],avgR2CbmRsm[f],

rhoMuRsmMuCbm[f],rhoNupRsmAlphaCbm[f],"\n")

clusterParms <- c(clusterParms, list(list(aucPro = aucPro, aucCbm = aucCbm, aucRsm = aucRsm,

nupRsm = nupRsm, alphaCbm = alphaCbm,

muRsm = muRsm, muCbm = muCbm)))

}else{

stop("Results file does not exist. Must analyze all datasets with mainRSM.R before running this.")

}

}

cat("\n")

cat(

mean(avgAucPro),mean(avgAucCbm),mean(avgAucRsm),

mean(avgSlopeProRsm),mean(avgR2ProRsm),

mean(avgSlopeCbmRsm),mean(avgR2CbmRsm),

mean(rhoMuRsmMuCbm),mean(rhoNupRsmAlphaCbm),"\n"

)

if (!file.exists("InterCorrelationBootstrapResults")){

# boostrap cluster code follows

names(clusterParms) <- fileNames

cl <- makeCluster(detectCores())

registerDoParallel(cl)

B <- 200

seed <- 1

bootStrapResults <- foreach (b = 1:B, .options.RNG = seed, .combine = "rbind", .packages = "RJafroc") %dorng% {

slopeCbmRsm <- rep(NA, length(fileNames));avgR2CbmRsm <- slopeCbmRsm

slopeProRsm <- slopeCbmRsm;avgR2ProRsm <- slopeCbmRsm

rhoMuRsmMuCbm <- rep(NA, length(fileNames));rhoNupRsmAlphaCbm <- rep(NA, length(fileNames))

AucRsm1 <- array(dim = c(length(fileNames)));AucPro1 <- AucRsm1;AucCbm1 <- AucRsm1

for (f in 1:length(fileNames)){

fileName <- fileNames[f]

retFileName <- paste0("allResults", fileName)

sysAnalFileName <- system.file("ANALYZED/RSM6", retFileName, package = "RJafroc")

if (file.exists(sysAnalFileName)){

load(sysAnalFileName)

I <- allResults[[1]]$I

J <- allResults[[1]]$J

AucRsm <- array(dim = c(I,J,length(fileNames)));AucPro <- AucRsm;AucCbm <- AucRsm

jBs <- ceiling(runif(J) \* J) # here is where we bootstap readers

AucRsm[,,f] <- clusterParms[[fileNames[f]]]$aucRsm[ , jBs]

AucPro[,,f] <- clusterParms[[fileNames[f]]]$aucPro[ , jBs]

AucCbm[,,f] <- clusterParms[[fileNames[f]]]$aucCbm[ , jBs]

AucRsm1[f] <- mean(AucRsm[,,f]);AucPro1[f] <- mean(AucPro[,,f]);AucCbm1[f] <- mean(AucCbm[,,f])

# constrained fit thru origin; aucPro vs. aucRsm

df1 <- data.frame(aucRsm = as.vector(clusterParms[[fileNames[f]]]$aucRsm[ , jBs]),

aucPro = as.vector(clusterParms[[fileNames[f]]]$aucPro[, jBs]))

mProRsm <- lm(aucPro ~ 0 + aucRsm, data = df1)

slopeProRsm[f] <- coef(mProRsm);avgR2ProRsm[f] <- summary(mProRsm)$r.squared

# constrained fit thru origin; aucCbm vs. aucRsm

df2 <- data.frame(aucRsm = as.vector(clusterParms[[fileNames[f]]]$aucRsm[ , jBs]),

aucCbm = as.vector(clusterParms[[fileNames[f]]]$aucCbm[, jBs]))

mCbmRsm <- lm(aucCbm ~ 0 + aucRsm, data = df2)

slopeCbmRsm[f] <- coef(mCbmRsm);avgR2CbmRsm[f] <- summary(mCbmRsm)$r.squared

# correlation between muRsm and muCbm

df1 <- data.frame(muRsm = as.vector(clusterParms[[fileNames[f]]]$muRsm[ , jBs]),

muCbm = as.vector(clusterParms[[fileNames[f]]]$muCbm[, jBs]))

rhoMuRsmMuCbm[f] <- cor(df1$muRsm, df1$muCbm,method = "pearson")

# correlation between nupRsm and alphaCbm

df2 <- data.frame(nupRsm = as.vector(clusterParms[[fileNames[f]]]$nupRsm[ , jBs]),

alphaCbm = as.vector(clusterParms[[fileNames[f]]]$alphaCbm[, jBs]))

rhoNupRsmAlphaCbm[f] <- cor(df2$nupRsm, df2$alphaCbm,method = "pearson")

}else{

stop("Results file does not exist.")

}

}

c(mean(AucPro1),mean(AucCbm1),mean(AucRsm1),

mean(slopeProRsm),mean(avgR2ProRsm),

mean(slopeCbmRsm),mean(avgR2CbmRsm),

mean(rhoMuRsmMuCbm),mean(rhoNupRsmAlphaCbm))

}

stopCluster(cl)

save(bootStrapResults, file = "InterCorrelationBootstrapResults")

} else load(file = "InterCorrelationBootstrapResults")

aucPro <- data.frame(value = bootStrapResults[ , 1])

aucCbm <- data.frame(value = bootStrapResults[ , 2])

aucRsm <- data.frame(value = bootStrapResults[ , 3])

slopeProRsm <- data.frame(value = bootStrapResults[ ,4])

#avgR2ProRsm <- data.frame(value = bootStrapResults[ ,5])

slopeCbmRsm <- data.frame(value = bootStrapResults[ ,6])

#avgR2CbmRsm <- data.frame(value = bootStrapResults[ ,7])

rhoMuRsmMuCbm <- data.frame(value = bootStrapResults[ ,8])

rhoNupRsmAlphaCbm <- data.frame(value = bootStrapResults[ ,9])

ciAvgAucPro <- quantile(aucPro$value, c(0.025, 0.975), type = 1)

cat("The empirical 95% CI of avgAucPro is", paste(ciAvgAucPro, collapse = ", "), "\n")

ciAvgAucCbm <- quantile(aucCbm$value, c(0.025, 0.975), type = 1)

cat("The empirical 95% CI of avgAucCbm is", paste(ciAvgAucCbm, collapse = ", "), "\n")

ciAvgAucRsm <- quantile(aucRsm$value, c(0.025, 0.975), type = 1)

cat("The empirical 95% CI of avgAucRsm is", paste(ciAvgAucRsm, collapse = ", "), "\n")

histogram <- ggplot(slopeProRsm, aes(x = value)) + geom\_histogram(color = "white") +

xlab("slopeProRsm") + scale\_color\_manual(values = "black") +

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

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

if (showPlot) print(histogram)

cislopeProRsm <- quantile(slopeProRsm$value, c(0.025, 0.975), type = 1)

cat("The empirical 95% CI of slopeProRsm is", paste(cislopeProRsm, collapse = ", "), "\n")

histogram <- ggplot(slopeCbmRsm, aes(x = value)) + geom\_histogram(color = "white") +

xlab("slopeCbmRsm") + scale\_color\_manual(values = "black") +

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

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

if (showPlot) print(histogram)

cislopeCbmRsm <- quantile(slopeCbmRsm$value, c(0.025, 0.975), type = 1)

cat("The empirical 95% CI of slopeCbmRsm is", paste(cislopeCbmRsm, collapse = ", "), "\n")

histogram <- ggplot(rhoMuRsmMuCbm, aes(x = value)) +

geom\_histogram(color = "white") +

xlab("rhoMuRsmMuCbm") + scale\_color\_manual(values = "black") +

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

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

if (showPlot) print(histogram)

ciRhoMuRsmMuCbm <- quantile(rhoMuRsmMuCbm$value, c(0.025, 0.975), type = 1)

cat("The empirical 95% CI of rhoMuRsmMuCbm is", paste(ciRhoMuRsmMuCbm, collapse = ", "), "\n")

histogram <- ggplot(rhoNupRsmAlphaCbm, aes(x = value)) +

geom\_histogram(color = "white") +

xlab("rhoNupRsmAlphaCbm") + scale\_color\_manual(values = "black") +

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

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

if (showPlot) print(histogram)

ciRhoNupRsmAlphaCbm <- quantile(rhoNupRsmAlphaCbm$value, c(0.025, 0.975), type = 1)

cat("The empirical 95% CI of rhoNupRsmAlphaCbm is", paste(ciRhoNupRsmAlphaCbm, collapse = ", "), "\n")

Ensure that line 29 is commented and click Source yielding the output shown below (if the showPlots flag is set to TRUE, one also sees plots shown in book Figures 19.6 and 19.7. [If errors occur, it may help to start a new session – Session, New Session and / or restart R – Session, Restart R.]

You may see messages in red-font, which are warnings or errors[[1]](#footnote-1), as in the following example:

Error in library(doParallel) : there is no package called ‘doParallel’

This means the package doParallel has not been downloaded. Use Packages, Install, etc. to download it.

## Online Appendix 19.B.2: Code output

> source(…)

0.812702 0.808441 0.817496 0.99421 0.999959 0.988724 0.999863 0.730396 0.970736

0.930201 0.925996 0.924517 1.00582 0.999782 1.00164 0.999975 0.591875 0.982425

0.861567 0.859958 0.862127 0.999338 0.999998 0.997526 0.999993 0.107013 0.918063

0.858656 0.846747 0.848447 1.01171 0.999787 0.997807 0.999942 0.839077 0.97499

0.891063 0.892147 0.894293 0.995756 0.999521 0.997561 0.999983 0.873189 0.96785

0.72518 0.696843 0.699174 1.03423 0.99849 0.996208 0.999944 0.981804 0.980522

0.856068 0.846301 0.842202 1.01578 0.999622 1.00442 0.999753 0.681472 0.830349

0.835685 0.827026 0.829238 1.00719 0.999635 0.997295 0.999965 0.860762 0.919168

0.85664 0.868027 0.869719 0.984741 0.999753 0.997918 0.99996 0.708928 0.864163

0.78772 0.783228 0.788529 0.99862 0.99992 0.99298 0.999885 0.798939 0.963516

0.729943 0.704451 0.714391 1.02005 0.998847 0.98383 0.998868 0.935813 0.908159

0.697161 0.673816 0.691341 1.00727 0.99846 0.973154 0.997549 0.94315 0.973108

0.635428 0.61472 0.61861 1.028 0.99801 0.991775 0.998413 0.96046 0.878034

0.883695 0.881001 0.88134 1.0027 0.999909 0.99962 1 0.993013 0.991371

0.811551 0.80205 0.805816 1.00753 0.999407 0.994319 0.999578 0.786135 0.937318

The empirical 95% CI of avgAucPro is 0.803327743790397, 0.819576348542326

The empirical 95% CI of avgAucCbm is 0.791793369296937, 0.808806314982641

The empirical 95% CI of avgAucRsm is 0.795487422212957, 0.812823905664956

The empirical 95% CI of slopeProRsm is 1.00655905539981, 1.01242541163394

The empirical 95% CI of slopeCbmRsm is 0.991757204528701, 0.996293969834235

The numbers were used to populate book Table 19.4.

## Online Appendix 19.B.3: Code explanation – non bootstrap

Line 13 defines the type of correlation to calculate as "pearson" (for Pearson, other choices are Kendall and Spearman[[2]](#footnote-2)) and showPlot as FALSE, which suppresses all plots. Line 14 – 18 defines the names of the datasets, which are identical to those used in mainRSM.R. Line 20 initializes clusterParms (for cluster parameters[[3]](#footnote-3)) to list(), an empty list. The parameters passed to the cluster code are "built up" by concatenating a list at line 123 - 125. Lines 21 – 27 assign a number of empty arrays to hold values for eventual averaging. For example, avgAucPro is a length-14 empty array, which is filled in, at line 110, by the average, over all readers and modalities, of the PROPROC AUC values for each dataset.

As usual, the best way to understand the code is to insert a break point and click Source (however, the breakpoint cannot be inside the cluster code). Insert a break point at line 33 and click Source. Lines 30 – 32 form the name of the pre-analyzed results file sysAnalFileName, similar to the procedure employed in mainRSM.R. For f = 1 the name is:

Browse[2]> sysAnalFileName

[1] "/Library/Frameworks/R.framework/Versions/3.4/Resources/library/RJafroc/ANALYZED/RSM6/allResultsTONY"

Clicking Next loads the contents of this file and the next two lines extract the number of modalities (2) are readers (5) in this dataset. The next few lines create a number of [I,J] dimensional arrays to hold the various results that pertain to this dataset. Advance the code pointer to line 44. Highlight the right hand side and click Run.

Browse[2]> allResults[[AllResIndx]]$retRsm$AUC

[1] 0.813194

This means that for the first modality, first reader of the "TONY" dataset, RSM-AUC is 0.813194. Click on "Execute the remainder of the current loop …" button twice to get out of the two for-loops. The code pointer should be at line 55. Highlight aucRsm at line 44 and click Run to reveal its contents. These are the RSM-fitted AUCs for this dataset:

Browse[2]> aucRsm

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

[1,] 0.813194 0.890863 0.852530 0.864690 0.895657

[2,] 0.678496 0.757884 0.792604 0.887828 0.741210

Feel free to print out values of other parameters: aucPro contains the PROPROC fitted AUCs, aucCbm contains the CBM fitted AUCs, muRsm contains the RSM  parameters, nupRsm contains the RSM  parameters, muCbm contains the CBM  parameters and alphaCbm contains the CBM  parameters.

Execute line 55 and reveal the contents of the data frame object:

Browse[2]> df

aucRsm aucPro

1 0.813194 0.801416

2 0.678496 0.671657

3 0.890863 0.894790

4 0.757884 0.754474

5 0.852530 0.852660

6 0.792604 0.793179

7 0.864690 0.857778

8 0.887828 0.874027

9 0.895657 0.890939

10 0.741210 0.736099

This is in a form that makes it easy to perform least squares fitting (line 56) and calculate the slope parameter (line 57) and the R2 of the fit (line 58). Execute these lines and print out the values:

Browse[2]> mProRsm

Call:

lm(formula = aucPro ~ 0 + aucRsm, data = df)

Coefficients:

aucRsm

0.994

Browse[2]> avgSlopeProRsm[f]

[1] 0.99421

Browse[2]> avgR2ProRsm[f]

[1] 0.999959

Notice the zero intercept constraint applied in the call to lm(). The slope of the PROPROC vs. RSM AUC values is 0.99421 and the R2 is 0.999959. These values are saved to arrays indexed by f (the dataset number). Lines 59 – 68 implement the scatter plot of aucPro vs. aucRsm and the zero-intercept fitted line, Figure 5.

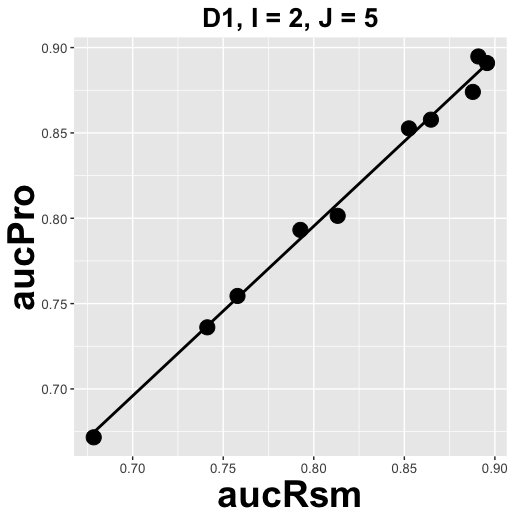


Figure 5: Plot created by command at line 68 (highlight p and click Run to see it).

Similarly, lines 71 – 83 implement the constrained fit thru origin of aucCbm vs. aucRsm and the scatter plot, Figure 6.

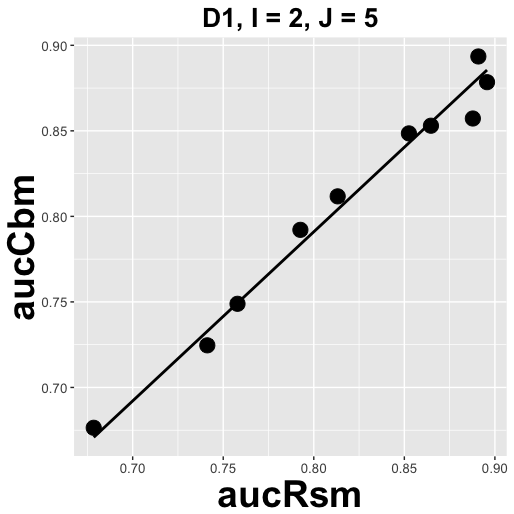


Figure 6: Plot created by command at line 83 (highlight p and click Run to see it).

Similarly, lines 85 – 95 implement the constrained fit thru origin of muCbm vs. muRsm and the scatter plot, Figure 7.

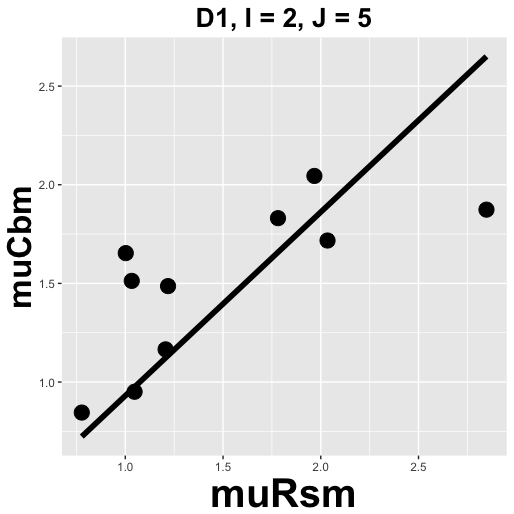


Figure 7: Plot created by command at line 95 (highlight p and click Run to see it).

Similarly, lines 97 – 107 implement the constrained fit thru origin of alphaCbm vs. nupRsm and the scatter plot, Figure 8.

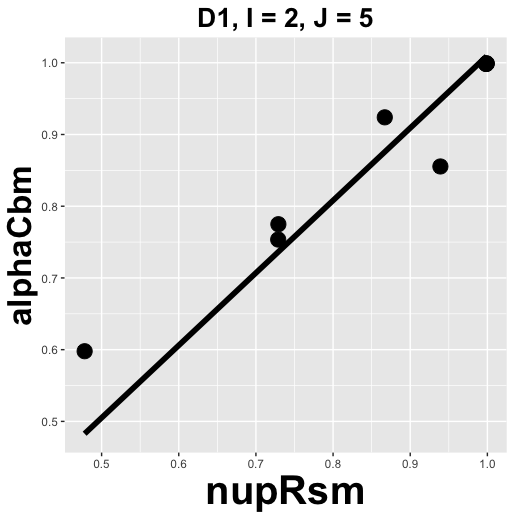


Figure 8: Plot created by command at line 107 (highlight p and click Run to see it).

The code pointer should be at line 109. The next three lines average aucPro, aucCbm and aucRsm, over modalities and readers and save the values to arrays indexed by f. Line 113 and 114 calculate the correlations rhoMuRsmMuCbm, i.e.,  and rhoNupRsmAlphaCbm, i.e., . For the first dataset the value are listed below.

Browse[2]> avgAucPro[f]

[1] 0.812702

Browse[2]> avgAucCbm[f]

[1] 0.808441

Browse[2]> avgAucRsm[f]

[1] 0.817496

Browse[2]> rhoMuRsmMuCbm[f]

[1] 0.730396

Browse[2]> rhoNupRsmAlphaCbm[f]

[1] 0.970736

Notice the near equality of the 3 AUC values and the relatively high values of the two correlations. [But this is for one dataset. To determine confidence intervals, a random reader bootstrap analysis, which averages over all datasets, is implemented in the code.] Line 116 – 119 prints out the first line of the output, listed above. Line 122 – 124 adds these values to the variable clusterParms that is passed to the cluster.

## Online Appendix 19.B.4: Code explanation – bootstrap part

Exit debug mode, clear all breakpoints (Session, Clear All Breakpoints, Yes) insert a break point at line 136 and click Source. Since the file InterCorrelationBootstrapResults exists (see File panel) the bootstrap code is not entered. Instead, clicking Next takes the code pointer to line 201 and a new variable, bootStrapResults, appears in the Environment panel. This is described below.

The actual bootstrap resampling of readers (only) occurs at line 158. To debug the bootstrap code one needs to temporarily rename the InterCorrelationBootstrapResults file, replace the for-each statement with a standard for-loop, etc. See file mainInterCorrelationsEasyBsDebug.R (for easy debugging of bootstrap code) for a demonstration of how this is done. In this code the cluster code has been replaced by standard code, the number of bootstraps B has been reduced to 20. Insert a break point at line 139, source the code and click next repeatedly to advance the code pointer to line 154. This line resamples readers for dataset f = 1:

Browse[2]> jBs

[1] 2 2 2 4 4

This is a 5 reader dataset, and this particular bootstrap sample picked reader 2 three times and reader 4 two times. Notice the difference in line 198, where 9 values are assigned to bootStrapResults[b,].

Cluster computing methods are used to speed up the bootstrap computations. Use the help files if you wish to understand more about cluster computing. Lines 140 and 141 are needed before one can run the cluster code.

cl <- makeCluster(detectCores())

registerDoParallel(cl)

The actual cluster code begins at line 144 with the for.each statement and ends with the closing parenthesis at line 197.

bootStrapResults <- foreach (b = 1:B, .options.RNG = seed, .combine = "rbind", .packages = "RJafroc") %dorng% {

…

}

The cluster code ends with stopCluster(cl) at line 198.

bootStrapResults is a [200,9] matrix, built up, row by row, at line 192 - 195. The 200 comes from the number of bootstraps, and the 9 comes from the number of elements in the array defined at line 192 – 195, *implicitly assigned to the left hand side of line 143*, i.e., to bootStrapResults.

The values are extracted and converted to data frames at lines 201 – 209. Line 201 extracts 200 bootstrap PROPROC AUC values as a data frame; line 202 extracts 200 bootstrap CBM AUC values; line 203 extracts 200 bootstrap RSM AUC values; line 204 extracts 200 bootstrap RSM vs. PROPROC slopes; line 206 extracts 200 bootstrap RSM vs. CBM slopes; line 208 extracts 200 bootstrap  values; and line 209 extracts 200 bootstrap  values. The bootstrap values are used to calculate 95% confidence intervals, using the quantile function, at lines 211, 214 and 217, etc. and to plot histograms.

# Online Appendix 19.C: Intra-correlations

This deals with correlations between three measures of performance predicted by the RSM: search performance S, lesion classification performance C and AUC performance A. The code is in mainIntraCorrelations.R, a listing of which follows.

## Online Appendix 19.C.1: Code Listing

# mainIntraCorrelations.R

# Fig. 19.8d,e,f,g,h,i

rm(list = ls())

library(foreach)

library(RJafroc)

library(doRNG)

library(doParallel)

library(ggplot2)

type <- "pearson";showPlot <- FALSE;showResultOneDataset <- FALSE

fileNames <- c("TONY", "VD", "FR",

"FED", "JT", "MAG",

"OPT", "PEN", "NICO",

"RUS", "DOB1", "DOB2",

"DOB3", "FZR")

avgS <- array(dim = length(fileNames))

avgC <-avgS;avgA <- avgS;RhoSC <- avgS;RhoSA <- avgS;RhoAC <- avgS

clusterParms <- list()

for (f in 1:length(fileNames)){

if (showResultOneDataset) if (f != 11) next

fileName <- fileNames[f]

retFileName <- paste0("allResults", fileName)

sysAnalFileName <- system.file("ANALYZED/RSM6", retFileName, package = "RJafroc")

if (file.exists(sysAnalFileName)){

load(sysAnalFileName)

I <- allResults[[1]]$I

J <- allResults[[1]]$J

S <- array(dim = c(I, J));C <- S;A <- S

AllResIndx <- 0

for (i in 1:I){

for (j in 1:J){

AllResIndx <- AllResIndx + 1

mu <- allResults[[AllResIndx]]$retRsm$mu

lambdaP <- allResults[[AllResIndx]]$retRsm$lambdaP

nuP <- allResults[[AllResIndx]]$retRsm$nuP

S[i, j] <- nuP \* exp(-lambdaP)

C[i, j] <- pnorm(mu/sqrt(2))

A[i, j] <- allResults[[AllResIndx]]$retRsm$AUC

}

}

avgS[f] <- mean(S)

avgC[f] <- mean(C)

avgA[f] <- mean(A)

RhoSC[f] <- cor(as.vector(S), as.vector(C), method = type)

RhoAC[f] <- cor(as.vector(A), as.vector(C), method = type)

RhoSA[f] <- cor(as.vector(S), as.vector(A), method = type)

cat("f = ", f,

" S[f] =", avgS[f],

" C[f] =", avgC[f],

" A[f] =", avgA[f],

" RhoSC[f] =", RhoSC[f],

" RhoSA[f] =", RhoSA[f],

" RhoAC[f] =", RhoAC[f],

"\n")

df <- data.frame(S = as.vector(S), C = as.vector(C))

ij <- paste0("D", f, ", I = ", I, ", J = ", J)

p <- ggplot(data = df, aes(x = S, y = C)) +

geom\_smooth(method = "lm", se = FALSE, color = "black", formula = y ~ x, size = 2) +

geom\_point(size = 5) +

labs(title = ij) +

theme(plot.title = element\_text(hjust = 0.5, size = 20, face = "bold")) +

scale\_color\_manual(values = "black") +

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

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

if (showPlot) print(p)

if (showResultOneDataset) {

m <- lm(S ~ C, df);

cat(fileNames[f], " S vs. C slope = ", as.numeric(m$coefficients[2]), ", r2 = ", summary(m)$r.squared,"\n")

}

df <- data.frame(S = as.vector(S), A = as.vector(A))

ij <- paste0("D", f, ", I = ", I, ", J = ", J)

p <- ggplot(data = df, aes(x = S, y = A)) +

geom\_smooth(method = "lm", se = FALSE, color = "black", formula = y ~ x, size = 2) +

geom\_point(size = 5) +

labs(title = ij) +

theme(plot.title = element\_text(hjust = 0.5, size = 20, face = "bold")) +

scale\_color\_manual(values = "black") +

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

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

if (showPlot) print(p)

if (showResultOneDataset) {

m <- lm(S ~ A, df);

cat(fileNames[f], " S vs. A slope = ", as.numeric(m$coefficients[2]), ", r2 = ", summary(m)$r.squared,"\n")

}

df <- data.frame(C = as.vector(C), A = as.vector(A))

ij <- paste0("D", f, ", I = ", I, ", J = ", J)

p <- ggplot(data = df, aes(x = C, y = A)) +

geom\_smooth(method = "lm", se = FALSE, color = "black", formula = y ~ x, size = 2) +

geom\_point(size = 5) +

labs(title = ij) +

theme(plot.title = element\_text(hjust = 0.5, size = 20, face = "bold")) +

scale\_color\_manual(values = "black") +

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

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

if (showPlot) print(p)

if (showResultOneDataset) {

m <- lm(C ~ A, df);

cat(fileNames[f], " C vs. A slope = ", as.numeric(m$coefficients[2]), ", r2 = ", summary(m)$r.squared,"\n")

}

clusterParms <- c(clusterParms, list(list(S = S, C = C, A = A)))

}else{

stop("Results file does not exist.")

}

}

if (showResultOneDataset) stop("stop for one dataset analysis")

cat(

"Avg S =", mean(avgS),"\n",

"Avg C =", mean(avgC),"\n",

"Avg A =", mean(avgA),"\n",

"Avg rhoSC =", mean(RhoSC),"\n",

"Avg rhoSA =", mean(RhoSA),"\n",

"Avg rhoAC =", mean(RhoAC),"\n",

"\n")

names(clusterParms) <- fileNames

if (!file.exists("IntraCorrelationBootstrapResults")){

cl <- makeCluster(detectCores())

registerDoParallel(cl)

B <- 200;seed <- 1

rho <- foreach (b = 1:B, .options.RNG = seed, .combine = "rbind", .packages = "RJafroc") %dorng% {

rhoSCb <- rep(NA, length(fileNames));rhoACb <- rhoSCb;rhoSAb <- rhoSCb

Sb1 <- array(dim = c(length(fileNames)));Cb1 <- Sb1;Ab1 <- Sb1

for (f in 1:length(fileNames)){

fileName <- fileNames[f]

retFileName <- paste0("allResults", fileName)

sysAnalFileName <- system.file("ANALYZED/RSM6", retFileName, package = "RJafroc")

if (file.exists(sysAnalFileName)){

load(sysAnalFileName)

I <- length(clusterParms[[fileNames[f]]]$S[,1])

J <- length(clusterParms[[fileNames[f]]]$S[1,])

Sb <- array(dim = c(I,J,length(fileNames)));Cb <- Sb;Ab <- Sb

jBs <- ceiling(runif(J) \* J) # bootstrap readers

Sb[,,f] <- clusterParms[[fileNames[f]]]$S[ , jBs]

Cb[,,f] <- clusterParms[[fileNames[f]]]$C[ , jBs]

Ab[,,f] <- clusterParms[[fileNames[f]]]$A[ , jBs]

Sb1[f] <- mean(Sb[,,f]);Cb1[f] <- mean(Cb[,,f]);Ab1[f] <- mean(Ab[,,f])

rhoSCb[f] <- cor(as.vector(Sb[,,f]), as.vector(Cb[,,f]), method = type)

rhoACb[f] <- cor(as.vector(Ab[,,f]), as.vector(Cb[,,f]), method = type)

rhoSAb[f] <- cor(as.vector(Sb[,,f]), as.vector(Ab[,,f]), method = type)

}else{

stop("Results file does not exist.")

}

}

c(mean(rhoSCb), mean(rhoACb), mean(rhoSAb), mean(Sb1), mean(Cb1), mean(Ab1))

}

stopCluster(cl)

save(bootStrapResults, file = "IntraCorrelationBootstrapResults")

} else load(file = "IntraCorrelationBootstrapResults")

rhoSCb <- data.frame(value = bootStrapResults[ , 1])

rhoACb <- data.frame(value = bootStrapResults[ , 2])

rhoSAb <- data.frame(value = bootStrapResults[ , 3])

Sb <- data.frame(value = bootStrapResults[ , 4])

Cb <- data.frame(value = bootStrapResults[ , 5])

Ab <- data.frame(value = bootStrapResults[ , 6])

histogram <- ggplot(Sb, aes(x = value)) +

geom\_histogram(color = "white") +

xlab("S") +

theme(plot.title = element\_text(hjust = 0.5, size = 20, face = "bold")) +

scale\_color\_manual(values = "black") +

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

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

print(histogram)

ciS <- quantile(Sb$value, c(0.025, 0.975), type = 1)

cat("The empirical 95% CI of S is", paste(ciS, collapse = ", "), "\n")

histogram <- ggplot(Cb, aes(x = value)) +

geom\_histogram(color = "white") +

xlab("C") +

theme(plot.title = element\_text(hjust = 0.5, size = 20, face = "bold")) +

scale\_color\_manual(values = "black") +

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

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

print(histogram)

ciC <- quantile(Cb$value, c(0.025, 0.975), type = 1)

cat("The empirical 95% CI of C is", paste(ciC, collapse = ", "), "\n")

histogram <- ggplot(Ab, aes(x = value)) +

geom\_histogram(color = "white") +

xlab("A") +

theme(plot.title = element\_text(hjust = 0.5, size = 20, face = "bold")) +

scale\_color\_manual(values = "black") +

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

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

print(histogram)

ciA <- quantile(Ab$value, c(0.025, 0.975), type = 1)

cat("The empirical 95% CI of A is", paste(ciA, collapse = ", "), "\n")

histogram <- ggplot(rhoSCb, aes(x = value)) +

geom\_histogram(color = "white") +

xlab("rho(S,C)") +

theme(plot.title = element\_text(hjust = 0.5, size = 20, face = "bold")) +

scale\_color\_manual(values = "black") +

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

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

print(histogram)

ciSC <- quantile(rhoSCb$value, c(0.025, 0.975), type = 1)

cat("The empirical 95% CI of the correlation between S and C is", paste(ciSC, collapse = ", "), "\n")

histogram <- ggplot(rhoACb, aes(x = value)) +

geom\_histogram(color = "white") +

xlab("rho(A,C)") +

theme(plot.title = element\_text(hjust = 0.5, size = 20, face = "bold")) +

scale\_color\_manual(values = "black") +

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

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

print(histogram)

ciAC <- quantile(rhoACb$value, c(0.025, 0.975), type = 1)

cat("The empirical 95% CI of the correlation between A and C is", paste(ciAC, collapse = ", "), "\n")

histogram <- ggplot(rhoSAb, aes(x = value)) +

geom\_histogram(color = "white") +

xlab("rho(S,A)") +

theme(plot.title = element\_text(hjust = 0.5, size = 20, face = "bold")) +

scale\_color\_manual(values = "black") +

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

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

print(histogram)

ciSA <- quantile(rhoSAb$value, c(0.025, 0.975), type = 1)

cat("The empirical 95% CI of the correlation between S and A is", paste(ciSA, collapse = ", "), "\n")

Sourcing the code yields the following output (warnings are not shown).

## Online Appendix 19.C.2: Code output

> source(…)

f = 1 S[f] = 0.254 C[f] = 0.833 A[f] = 0.817 RhoSC[f] = -0.394 RhoSA[f] = 0.636 RhoAC[f] = 0.246

f = 2 S[f] = 0.356 C[f] = 0.986 A[f] = 0.925 RhoSC[f] = -0.246 RhoSA[f] = 0.479 RhoAC[f] = 0.4

f = 3 S[f] = 0.0835 C[f] = 0.97 A[f] = 0.862 RhoSC[f] = -0.815 RhoSA[f] = 0.601 RhoAC[f] = -0.503

f = 4 S[f] = 0.266 C[f] = 0.931 A[f] = 0.848 RhoSC[f] = -0.681 RhoSA[f] = 0.112 RhoAC[f] = -0.417

f = 5 S[f] = 0.189 C[f] = 0.947 A[f] = 0.894 RhoSC[f] = -0.388 RhoSA[f] = 0.124 RhoAC[f] = 0.247

f = 6 S[f] = 0.0912 C[f] = 0.928 A[f] = 0.699 RhoSC[f] = -0.384 RhoSA[f] = 0.336 RhoAC[f] = -0.49

f = 7 S[f] = 0.312 C[f] = 0.917 A[f] = 0.842 RhoSC[f] = -0.333 RhoSA[f] = 0.391 RhoAC[f] = 0.296

f = 8 S[f] = 0.16 C[f] = 0.968 A[f] = 0.829 RhoSC[f] = -0.311 RhoSA[f] = 0.0735 RhoAC[f] = 0.203

f = 9 S[f] = 0.139 C[f] = 0.947 A[f] = 0.87 RhoSC[f] = -0.692 RhoSA[f] = -0.349 RhoAC[f] = 0.416

f = 10 S[f] = 0.15 C[f] = 0.858 A[f] = 0.789 RhoSC[f] = -0.616 RhoSA[f] = 0.215 RhoAC[f] = 0.243

f = 11 S[f] = 0.11 C[f] = 0.766 A[f] = 0.714 RhoSC[f] = -0.65 RhoSA[f] = 0.787 RhoAC[f] = -0.493

f = 12 S[f] = 0.215 C[f] = 0.764 A[f] = 0.691 RhoSC[f] = -0.481 RhoSA[f] = 0.581 RhoAC[f] = -0.0782

f = 13 S[f] = 0.0483 C[f] = 0.786 A[f] = 0.619 RhoSC[f] = -0.205 RhoSA[f] = 0.769 RhoAC[f] = -0.0461

f = 14 S[f] = 0.15 C[f] = 0.969 A[f] = 0.881 RhoSC[f] = -0.438 RhoSA[f] = -0.445 RhoAC[f] = 0.58

Avg S = 0.18

Avg C = 0.898

Avg A = 0.806

Avg rhoSC = -0.474

Avg rhoSA = 0.308

Avg rhoAC = 0.043

The empirical 95% CI of S is 0.160438768820473, 0.21847576065802

The empirical 95% CI of C is 0.876575909106635, 0.906981144409909

The empirical 95% CI of A is 0.795487422212957, 0.812823905664956

The empirical 95% CI of the correlation between S and C is -0.608099630403152, -0.350152055879301

The empirical 95% CI of the correlation between A and C is -0.113984659834585, 0.265965649964626

The empirical 95% CI of the correlation between S and A is 0.125106916214022, 0.437283986928133

## Online Appendix 19.C.3: Code explanation - non-bootstrap part

The code is similar to that in Online Appendix 19.B with some important differences: only RSM parameters mu, lambdaP, nuP are extracted (lines 35 - 37), S, C are calculated from these parameters and saved to IxJ dimensional arrays (lines 38 – 39) and A (RSM-AUC) is extracted to an IxJ dimensional array at line 40. Line 44 – 46 averages S, C and A over all modalities and readers in the dataset defined by f. Line 48 – 50 computes the correlations between S and C, A and C, and A and S, for the same dataset. The following statements print these values.

Insert a break point at line 70 and click Source. Highlight p and click Run, Figure 9.

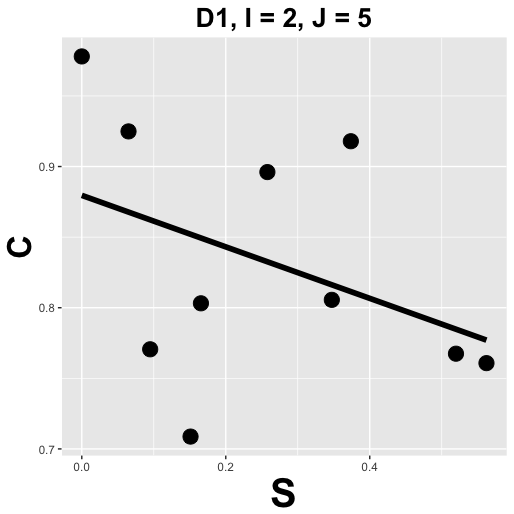


Figure 9: This figure was obtained by displaying the plot at line 70. Despite noise, there is evidence of an inverse correlation between search and lesion classification performance, for the first dataset. This is borne out by the subsequent bootstrap analysis.

Exit debug mode, clear the break point at line 70, insert a break point at line 86 and click Source. Highlight p and click Run, Figure 10.

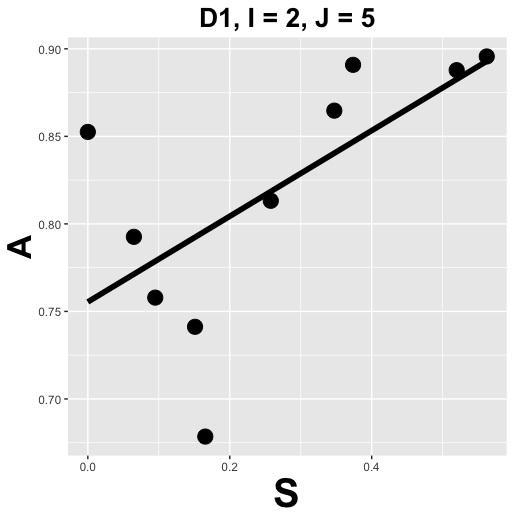


Figure 10: This figure was obtained by displaying the plot at line 86. There is evidence of a positive correlation between search and AUC performance, for the first dataset. This is also borne out by the subsequent bootstrap analysis.

Exit debug mode, clear the break point at line 86, insert a break point at line 102 and click Source. Highlight p and click Run, Figure 11.

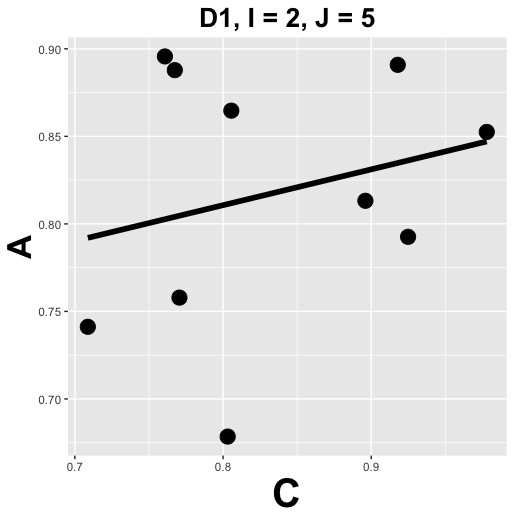


Figure 11: This figure was obtained by displaying the plot at line 102. While there is evidence of a weak positive correlation between AUC performance and lesion classification performance, for the first dataset, subsequent bootstrap analysis does not support this conclusion.

## Online Appendix 19.C.3: Code explanation - bootstrap part

At line 108 S, C and A are added to clusterParms for subsequent bootstrap analysis. Exit debug mode, remove any existing break point, insert a break point at line 125 and click Source. Since the file IntraCorrelationBootstrapResults already exists as a disk file, the bootstrap code between lines 126 and 158 is not entered. Instead the pre-analyzed results are loaded and a new object bootStrapResults appears in the Environment panel. The code pointer is at line 161. At lines 161 - 166, the contents of the new object are extracted into 200-dimensional arrays in the following order: rhoSCb, rhoACb, rhoSAb, Sb, Cb and Ab. The trailing b stands for bootstrap values.

## Online Appendix 19.C.4: PROPROC's degeneracy problem

This relates to book chapter 20.7.2, which discusses PROPROC's problem with fitting degenerate datasets.

Line 168 – 177 displays a histogram of the 200 values for search, Sb, calculates a 95% confidence interval and prints it, Figure 12.

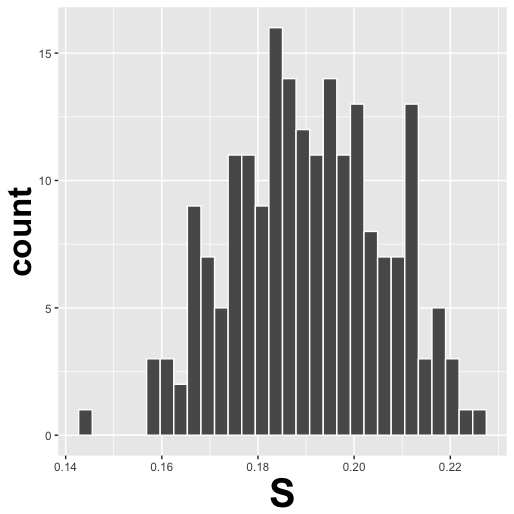


Figure 12: Histogram: seacrh performance, S. The empirical 95% CI of S is (0.16, 0.22).

Line 179 – 188 displays a histogram of the 200 values for lesion classification performance, Cb, calculates a 95% confidence interval and prints it, Figure 13.

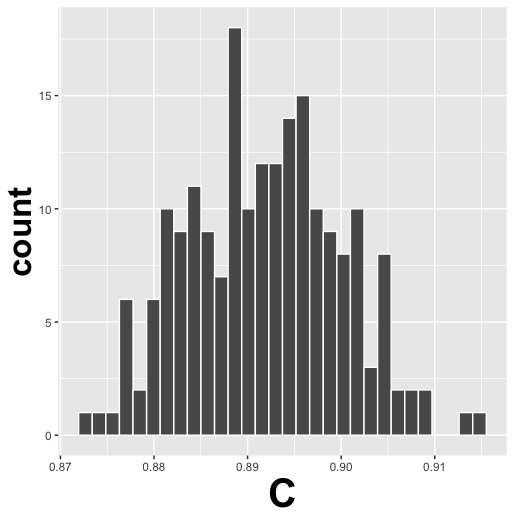


Figure 13: Histogram: lesion classification performance, C. The empirical 95% CI of C is (0.88, 0.91).

Line 190 – 199 displays a histogram of the 200 values for AUC performance, Ab, calculates a 95% confidence interval and prints it, Figure 14.

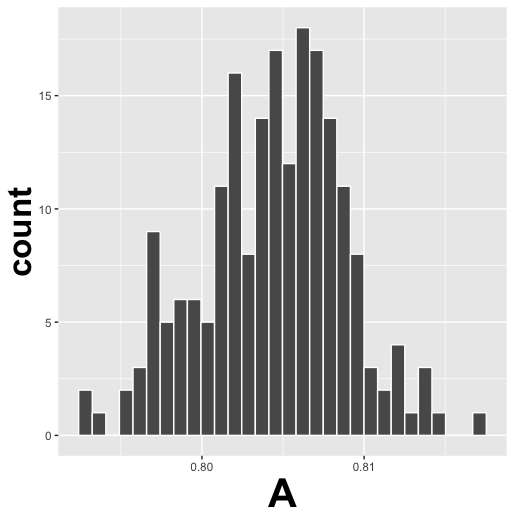


Figure 14: Histogram: AUC performance, A. The empirical 95% CI of A is (0.80, 0.81).

Line 201 – 210 displays a histogram of the 200 values of the correlation between Sb and Cb, rhoSCb, calculates a 95% confidence interval and prints it, Figure 15.

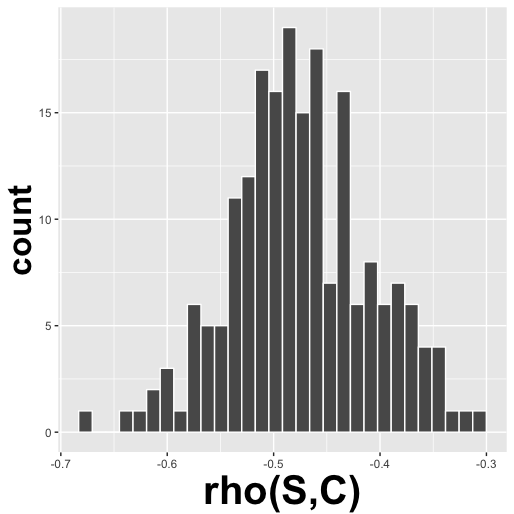


Figure : Histogram: correlation beween S and C. The empirical 95% CI of the correlation between S and C is (-0.61, -0.35).

Line 212 – 221 displays a histogram of the 200 values of the correlation between Ab and Cb, rhoACb, calculates a 95% confidence interval and prints it, Figure 16.

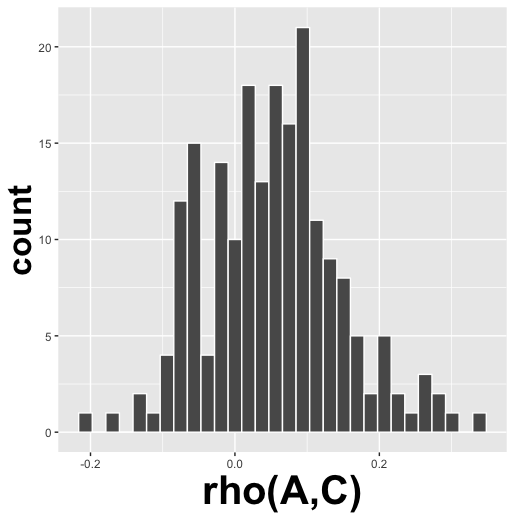


Figure 16: Histogram: correlation beween A and C. The empirical 95% CI of the correlation between A and C is (-0.11, 0.27).

Line 223 – 232 displays a histogram of the 200 values of the correlation between Sb and Ab, rhoSAb, calculates a 95% confidence interval and prints it, Figure 17.

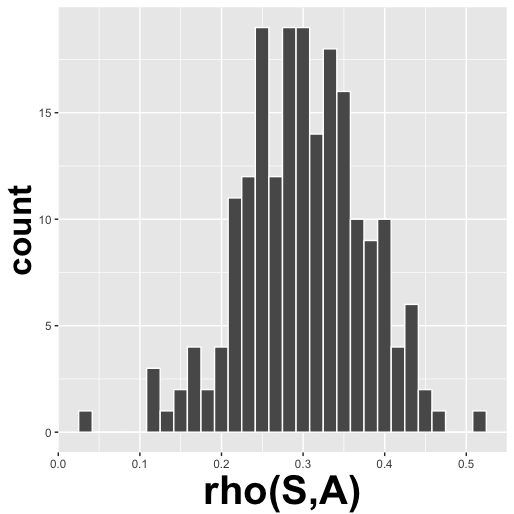


Figure 17: Histogram: correlation beween S and A. The empirical 95% CI of the correlation between S and A is (0.13, 0.44).

# Online Appendix 19.D: Sample size estimation for FROC studies

This appendix relates to book chapter 19.7. The following is a listing of code for sample size estimation using both inferred ROC-AUC and wAFROC-AUC as figures of merit. The code listing of the relevant file mainwAFROCPowerDBMH.R follows.

### Online Appendix 19.D.1: Code listing

#mainwAFROCPowerDBMH.R

rm(list = ls())

library(ggplot2)

library(RJafroc)

# included datasets

fileNames <- c("TONY", "VD", "FR",

"FED", "JT", "MAG",

"OPT", "PEN", "NICO",

"RUS", "DOB1", "DOB2",

"DOB3", "FZR")

f <- 4

fileName <- fileNames[f]

theData <- get(sprintf("dataset%02d", f))

# RSM ROC fitting needs to know lesionDistribution

nLesDistr <- UtilLesionDistribution(theData)

retFileName <- paste0("allResults", fileName)

sysAnalFileName <- system.file(

"ANALYZED/RSM6",

retFileName,

package = "RJafroc")

load(sysAnalFileName)

I <- allResults[[1]]$I

J <- allResults[[1]]$J

mu <- array(dim = c(I, J))

nuP <- array(dim = c(I, J))

lambdaP <- array(dim = c(I, J))

lambda <- array(dim = c(I, J))

nu <- array(dim = c(I, J))

AllResIndx <- 0

for (i in 1:I){

for (j in 1:J){

AllResIndx <- AllResIndx + 1

mu[i,j] <- allResults[[AllResIndx]]$retRsm$mu

lambdaP[i,j] <- allResults[[AllResIndx]]$retRsm$lambdaP

nuP[i,j] <- allResults[[AllResIndx]]$retRsm$nuP

x <- UtilPhysical2IntrinsicRSM(

mu[i,j], lambdaP[i,j], nuP[i,j])

lambda[i,j] <- x$lambda

nu[i,j] <- x$nu

}

}

# FED data has 5 modalities; we choose to analyze the first two

i1 <- 1;i2 <- 2

cat("NH i1 = ", i1, "NH i2 = ", i2, "\n")

selectJ <- c(1, 2, 3, 4)

frocData <- DfExtractDataset(

theData, trts = c(i1, i2), rdrs = selectJ)

J <- length(frocData$readerID)

K <- dim(frocData$NL)[3]

mu1 <- mu[1,]

mu2 <- mu[2,]

nu1 <- nu[1,]

nu2 <- nu[2,]

lambda1 <- lambda[1,]

lambda2 <- lambda[2,]

mu <- rbind(mu1, mu2)

lambda <- rbind(lambda1, lambda2)

nu <- rbind(nu1, nu2)

# instead of average, use median to get representative value over whole dataset

muMed <- median(mu)

nuMed <- median(nu) # do:

lambdaMed <- median(lambda) # do:

# construct lesion weights, assuming equally weighted lesions

lesionWeights <- matrix(-Inf, nrow = nrow(nLesDistr), ncol = nrow(nLesDistr))

for (l in 1:nrow(nLesDistr)){

nLes <- nLesDistr[l, 1]

lesionWeights[l, 1:nLes] <- 1/nLes

}

# calculate NH values for ROC-AUC and wAFROC-AUC

aucRocNH <- PlotRsmOperatingCharacteristics(

muMed, lambdaMed, nuMed,

lesionDistribution = nLesDistr, lesionWeights = lesionWeights)$aucROC

aucAfrocNH <- PlotRsmOperatingCharacteristics(

muMed, lambdaMed, nuMed,

lesionDistribution = nLesDistr, lesionWeights = lesionWeights)$aucwAFROC

# following code calculates ROC-ES and wAFROC-ES

deltaMu <- seq(0.01, 0.2, 0.01) # values of deltaMu to scan below

esRoc <- array(dim = length(deltaMu))

eswAfroc <- array(dim = length(deltaMu))

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

esRoc[i] <- PlotRsmOperatingCharacteristics(

muMed + deltaMu[i],

lambdaMed, nuMed,

lesionDistribution = nLesDistr,

lesionWeights = lesionWeights,

type = "ROC")$aucROC - aucRocNH

eswAfroc[i] <- PlotRsmOperatingCharacteristics(

muMed+ deltaMu[i],

lambdaMed,

nuMed,

lesionDistribution = nLesDistr,

lesionWeights = lesionWeights,

type = "wAFROC")$aucwAFROC - aucAfrocNH

cat("ES ROC, wAFROC = ", esRoc[i], eswAfroc[i],"\n")

}

cat("\n")

a<-lm(eswAfroc~ 0+esRoc) # fit values to straight line thru origin

effectSizeROC <- seq(0.01, 0.1, 0.001)

effectSizewAFROC <- effectSizeROC\*a$coefficients[1]# r2 = summary(a)$r.squared

JTest <- 5;KTest <- 100

varCompROC <- StSignificanceTesting(

frocData,

FOM = "HrAuc",

method = "DBMH",

option = "RRRC")$varComp

varCompwAFROC <- StSignificanceTesting(

frocData,

FOM = "wAFROC",

method = "DBMH",

option = "RRRC")$varComp

cat("JTest = ", JTest, "KTest = ", KTest, "\n")

powerROC <- array(dim = length(effectSizeROC))

powerwAFROC <- array(dim = length(effectSizeROC))

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

varYTR <- varCompROC$varComp[3]

varYTC <- varCompROC$varComp[4]

varYEps <- varCompROC$varComp[6]

powerROC[i] <- SsPowerGivenJK(

JTest,

KTest,

alpha = 0.05,

effectSize = effectSizeROC[i],

option = "RRRC",

method = "DBMH",

varYTR = varYTR,

varYTC = varYTC,

varYEps = varYEps)$powerRRRC

varYTR <- varCompwAFROC$varComp[3]

varYTC <- varCompwAFROC$varComp[4]

varYEps <- varCompwAFROC$varComp[6]

powerwAFROC[i] <- SsPowerGivenJK(

JTest, KTest,

alpha = 0.05,

effectSize = effectSizewAFROC[i],

option = "RRRC",

method = "DBMH",

varYTR = varYTR,

varYTC = varYTC,

varYEps = varYEps)$powerRRRC

cat("ROC effect-size = ,", effectSizeROC[i],

"wAFROC effect-size = ,", effectSizewAFROC[i],

"Power ROC, wAFROC:", powerROC[i], ",", powerwAFROC[i], "\n")

}

df <- data.frame(esRoc = esRoc, eswAfroc = eswAfroc)

p <- ggplot(data = df, aes(x = esRoc, y = eswAfroc)) +

geom\_smooth(method = "lm", se = FALSE, color = "black", formula = y ~ x) +

geom\_point()

print(p)

df <- data.frame(powerROC = powerROC, powerwAFROC = powerwAFROC)

p <- ggplot(mapping = aes(x = powerROC, y = powerwAFROC)) +

geom\_line(data = df, size = 0.5)

print(p)

Line 13 defines the dataset to be loaded, namely the "FED" dataset, which is a 5-modality 4-radiologist FROC dataset. Place a break point at line 52 and click Source. Line 51 selects all four radiologists in the dataset and line 52-53 uses the ExtractDataset() function to extract data corresponding to the first two modalities only, as these are the ones regarded as NH modalities (or a pilot study that did not reveal a significant difference). Keep clicking on Next until the code pointer advances to line 54. Examination of the Environment panel reveals that now frocData is a 2-modality 4-reader dataset with 100 non-diseased and 100 diseased cases. Line 57-62 assigns memory to hold the parameters  where the appendage "1" applies to modality 1 and "2" applies to modality 2. Line 74 – 70 computes the median values for the three RSM parameters. Click Next to execute these statements. Line 73 – 77 calculates the lesion weights matrix, assuming equal weights for all lesions. Line 79 – 81 extracts the numerically integrated AUC under the RSM predicted ROC curve. Line 82 – 84 does the same for the AFROC AUC. Both of these quantities are calculated for the NH condition. Advance the code pointer to line 87, which creates a sequence vector deltaMu ranging from 0.01 to 0.2, with spacing 0.01. The idea here is that we will create a number of AH conditions using each of these values to increment muMed (the NH , i.e., the median of  over all readers and modalities in the NH condition). As noted in previous chapter, one can change AUC from 0.5 to 1 by simply incrementing .

Browse[2]> deltaMu

[1] 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.10 0.11 0.12 0.13 0.14 0.15 0.16 0.17 0.18 0.19

[20] 0.20

Line 88 – 89 assigns memory space for two vectors esRoc and eswAfroc, which will contain the effect sizes in ROC-AUC and wAFROC-AUC units. Lines 90 – 105 initializes these arrays. Advance the code pointer to line 106, obtaining the following output:

Browse[2]> f

ES ROC, wAFROC = 0.000625 0.00133

ES ROC, wAFROC = 0.00124 0.00265

ES ROC, wAFROC = 0.00186 0.00397

ES ROC, wAFROC = 0.00247 0.00527

ES ROC, wAFROC = 0.00307 0.00657

ES ROC, wAFROC = 0.00367 0.00786

ES ROC, wAFROC = 0.00426 0.00914

ES ROC, wAFROC = 0.00485 0.0104

ES ROC, wAFROC = 0.00544 0.0117

ES ROC, wAFROC = 0.00602 0.0129

ES ROC, wAFROC = 0.00659 0.0142

ES ROC, wAFROC = 0.00716 0.0154

ES ROC, wAFROC = 0.00772 0.0166

ES ROC, wAFROC = 0.00828 0.0179

ES ROC, wAFROC = 0.00884 0.0191

ES ROC, wAFROC = 0.00939 0.0203

ES ROC, wAFROC = 0.00994 0.0215

ES ROC, wAFROC = 0.0105 0.0227

ES ROC, wAFROC = 0.011 0.0239

ES ROC, wAFROC = 0.0115 0.0251

The code pointer should be at line 106. Line 108 fits a straight line constrained to go through the origin to these numbers. a$coefficients[1] contains the slope of this line, 2.16 in the current example. In other words, eswAfroc = 2.16 x esRoc. As stated in book chapter 19.7, this is the all-important equation that relates an ROC-AUC effect size that most researchers understand (due to its long-standing history) to a wAFROC-AUC effect size, whose history is more recent. Line 109 defines a finely spaced ROC effect size array ranging from 0.1 to 1. Line 110 converts it to a wAFROC effect size, essentially my multiplying the ROC effect size by 2.16. Line 112 assumes 5 readers and 100 cases in the pivotal study. Line 113 – 117 extracts the ROC analysis jackknife pseudovalue variance components.

Browse[2]> varCompROC

varComp

Var(R) 0.000828

Var(C) 0.038123

Var(T\*R) 0.000153

Var(T\*C) 0.009644

Var(R\*C) 0.003544

Var(Error) 0.094846

Line 113 – 117 extracts the wAFROC analysis jackknife pseudovalue variance components.

Browse[2]> varCompwAFROC

varComp

Var(R) 0.001854

Var(C) 0.061178

Var(T\*R) -0.000444

Var(T\*C) 0.010165

Var(R\*C) 0.013559

Var(Error) 0.096726

The variance components that affect sample size, namely Var(T\*R), Var(T\*C) and Var(Error), have comparable magnitudes. The main factor in determining the differences in sample sizes is the factor of 2.16 larger wAFROC effect size, which, since it appears as the square in the formula for the non-centrality parameter, results in substantially greater statistical wAFROC power, as compared to ROC, Figure 19.

The for-loop extending from line 128 – 158 computes power using both methods: lines 128 – 140 for ROC and lines 142 – 153 for wAFROC power, for each effect size specified in the ROC effect size array, and prints the results. Line 155 – 157 prints out these values and the rest of the code, outside the for-loop, displays plots, Figure 18 and Figure 19.

Note the consistent usage of ROC-AUC related quantities in line 128 – 140 and the consistent usage of wAFROC-AUC related quantities in line 142 - 153. This is the meaning of using the same "units" of AUC, mentioned in the book. The reason the wAFROC effect size is larger is that this AUC has a greater range of variation, 0 to 1, as compared to the ROC AUC, 0.5 to 1.

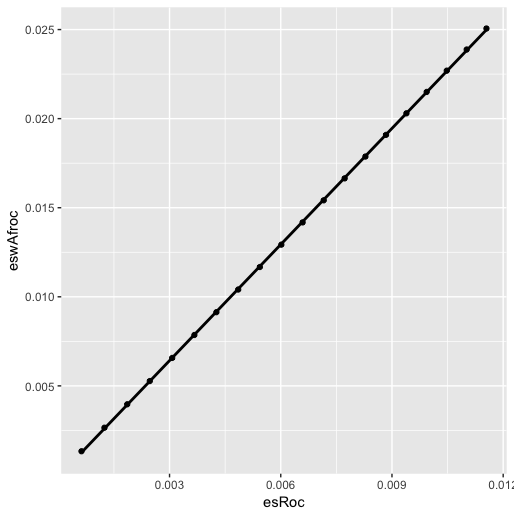


Figure 18: Plot of wAFROC effect size vs. ROC effect size. The slope is 2.16.

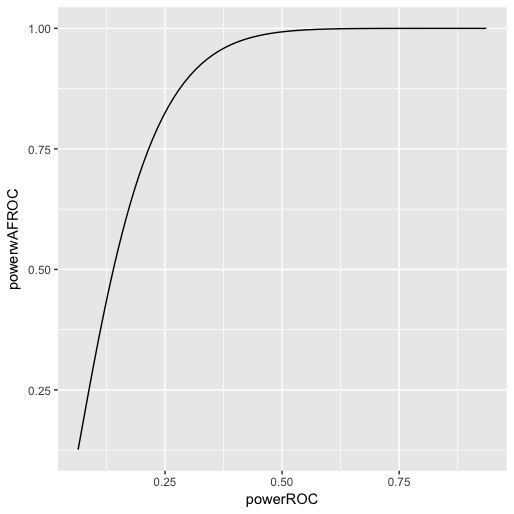


Figure 19: Plot of statistical power using wAFROC analysis vs. that using ROC analysis, for different equivalent effect sizes, i.e., each wAFROC effect size is 2.16 times the corresponding ROC effect size.

### Online Appendix 19.D.2: Code snippet

This section is left intentionally blank.

### Online Appendix 19.D.3: Code snippet

This section is left intentionally blank.

### Online Appendix 19.D.4: Code snippet

This section is left intentionally blank.

### Online Appendix 19.D.5: Code snippet

This section is left intentionally blank.

### Online Appendix 19.D.6: Code output

This section is left intentionally blank.

### Online Appendix 19.D.7: Code output

This section is left intentionally blank.

# References

1. Warnings can usually be disregarded, but errors cannot. [↑](#footnote-ref-1)
2. The correlations are calculated at lines 114, 115, 182 and 187, in each case the cor() function is used. [↑](#footnote-ref-2)
3. The cluster code, used to speed up the bootstrap computations, is in lines 138 - 199. As explained in the book, only readers are regarded as random. The code has already been run and the results saved to file InterCorrelationBootstrapResults; since this file exists, a file.exist test at line 137 ensures that the cluster code is not entered, and instead the precomputed results are loaded at line 200 using the load() function. If the user deletes, or preferably renames, the results file, then the cluster code will be entered, and after a while a new results file will be written to disk. It is difficult to debug inside the cluster code – the easiest way is to replace it with "normal" code – the author leaves this as an exercise to the user (basically one replaces the for.each with a for loop, and makes other changes). [↑](#footnote-ref-3)